IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of	:	**Confirmation No. 2956**
Tomoyuki OKADA et al.	:	**Docket No. PR03_0038A**
Serial No. 60/440,623	:	
Filed January 17, 2003	:	
DIGITAL IMAGE CONTENTS RECORDING MEDIUM, APPARATUS AND METHOD FOR REPRODUCING THE DIGITAL IMAGE CONTENTS	:	... COMMISSIONER IS AUTHORIZED ... CHARGE ANY DEFICIENCY IN THE ...EES FOR THIS PAPER TO DEPOSIT ...CCOUNT NO. 23-0975

SUBMISSION OF ENGLISH TRANSLATION WITH STATEMENT OF ACCURACY

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

The above-identified U.S. provisional patent application was filed on January 17, 2003 in a language other than English. Therefore, in accordance with 37 CFR 1.78(a)(5)(iv), an English language translation of the provisional application is submitted herewith together with a statement that the translation is accurate.

Respectfully submitted,

Tomoyuki OKADA et al.

Exhibit 4

By: _____
Jeffrey R. Filipek
Registration No. 41,471
Attorney for Applicants

JRF/fs
Washington, D.C. 20006-1021
Telephone (202) 721-8200
Facsimile (202) 721-8250
September 17, 2003

VERIFICATION OF TRANSLATION

I, Yukiko Nakayama, translator of Fushimi-ku, Kyoto, Japan, hereby declare that I am conversant with the English and Japanese languages and am a competent translator thereof. I further declare that to the best of my knowledge and belief the following is a true and correct translation made by me, of U.S. Provisional Application No. 60/440,623 filed on January 17, 2003.

Date: August 27, 2003

Yukiko Nakayama

Yukiko Nakayama

TITLE OF THE INVENTION

DIGITAL IMAGE CONTENTS RECORDING MEDIUM, APPARATUS AND METHOD FOR REPRODUCING THE DIGITAL IMAGE CONTENTS

BACKGROUND OF THE INVENTION

(1) Field of the Invention

The present invention relates to a medium on which digital image contents are recorded, and to an apparatus and a method for reproducing the digital image contents.

(2) Description of the Related Art

The following describes a DVD (hereafter referred to as an "SD-DVD" or simply a "DVD") that is a conventional technology.

FIG. 1 shows the structure of an SD-DVD. As shown in the lower part of FIG. 1, a DVD has a logical address space provided between the lead-in and the lead-out. At the head of the logical address space, volume information of a file system is recorded. Following the volume information, application data such as video and audio is recorded.

The file system referred to herein intends to mean the ISO9660 or the UDF (Universal Disc Format), which is a system expressing data on a disc in units of directories and files. A PC (personal computer) of daily use can present data stored in its hard disk in units of directories or files by employing

such a file system as the FAT (File Allocation Table) or the NTFS (New Technology File System). This enhances usability.

For an SD-DVD, both the UDF and the ISO9660 (a combination of the two is sometimes called "UDF Bridge") are employed. Therefore, data can be read from an SD-DVD using a file system driver for either the UDF or the ISO9660 (the DVD here is a ROM disc for package-media and therefore is not physically writable).

By employing the UDF Bridge, data recorded on a DVD can be viewed as directories or files shown in the upper left part of FIG. 1. Immediately below a root directory ("ROOT" in the figure), a directory called "VIDEO_TS" is positioned in which application data for the DVD is stored. The application data is recorded as a plurality of files. Main files include:

VIDEO_TS.IFO (disc playback control information file)

VTS_01_0.IFO (video title set#1 playback control information file)

VTS_01_0.VOB (video title set#1 stream file)

....

Each of these files is provided with one of two types of extensions. The extension "IFO" indicates that the file stores playback control information, whereas the extension

"VOB" indicates that the file stores an MPEG stream that is AV data. The playback control information includes information that is used to realize interactivity adopted for DVDs (the ability of dynamically changing the playback (reproduction) state in response to user operations), and information such as metadata that accompanies titles, AV streams, or the like. In DVDs, such playback control information can be commonly called "navigation information".

Provided as playback control information files are the "VIDEO_TS.IFO" that is used to manage the disc overall and the "VTS_01_0.IFO" that is playback control information provided for an individual video title set (note that a plurality of titles, i.e., different movies or movies of different versions, can be recorded on a single DVD). Here, "01" in the body of the file name indicates the number given to the video title set. For example, a playback control information file provided for a video title set#2 is "VTS_02_0.IFO".

The upper right part of FIG. 1 shows a DVD navigation space in the application layer of the DVD. This space has a logical structure in which the above-described playback control information is expanded. In the DVD navigation space, information written in the "VIDEO_TS.IFO" is expanded as VMGI (VIDEO Manager Information), and playback control information written in the "VTS_01_0.IFO" or in other files provided for

other video title sets is expanded as VTSI (Video Title Set Information).

The VTSI includes PGCI (Program Chain Information) that is information about a playback sequence called a "PGC (Program Chain)". The PGCI is made up of a plurality of cells (Cells) and a kind of programming information called a "command". Each cell (Cell) is made up of some sections or all sections of a VOB (a Video Object representing an MPEG stream). Playback of a cell is an equivalent to playback of sections of a VOB that are designated by the cell.

A command here is processed by a DVD virtual machine, and is similar to JavaScript or the like executed on a browser. The DVD command differs from JavaScript in the following point. Besides logical calculation, the DVD command simply provides playback control of an AV title. One example of such playback control is a designation of a chapter to be played back. On the other hand, JavaScript provides control of a window and a browser (e.g., opening a window of a new browser) besides logical calculation.

A cell has, as its internal information, start and end addresses (logical addresses) of a VOB that is recorded on the disc. A player reads data using the information about the start and end addresses of the VOB shown by the cell, and realizes playback of the data.

FIG. 1 is a schematic diagram for describing navigation

information embedded in an AV stream. The interactivity that is a feature of an SD-DVD is realized not only by navigation information written in the above-described "VIDEO_TS.IFO" and "VTS_01_0.IFO", but also by other several types of important information multiplexed together with video and audio data within a VOB using a dedicated carrier called a navigation pack (referred to as a "navi-pack" or an "NV-PCK").

The following describes a menu as a simple example of interactivity. Several buttons appear on a menu screen. Each button has the definition of processing to be executed when the button is selected. On the menu screen, one button is presently selected (the selected button is overlaid by a transparent highlight color), and the user can move the highlight color to buttons positioned up, down, left, and right of the presently selected button, using "up", "down", "left", and "right" arrow keys provided on a remote controller. The user moves the highlight color to a button that he or she wants to select, using the "up", "down", "left", and "right" arrow keys provided on the remote controller, and determines the selection (by pressing a "return" key). Then, a program of a command corresponding to the selected button is executed. Typically, playback of the corresponding title or chapter is executed by the command.

The upper left part of FIG. 2 gives a brief description of an NV_PCK.

The NV_PCK includes highlight color information and button information. The highlight color information includes information about a color palette, with which a transparent highlight color to be displayed as overlaid can be designated. The button information is provided for each button and includes rectangular area information showing a position of a rectangular area of the button, highlight move information showing the movement of a highlight color from the button to another button (information about buttons to which the highlight color is to be moved when the user presses the "up", "down", "left", and "right" arrow keys), and button command information (showing a command to be executed when the button selection is determined).

A highlight color on the menu screen is realized by an overlay image shown in the upper central right part of FIG. 2. The overlay image is made by painting a rectangular area shown by the rectangular area information for the selected button with a color selected from the color palette. This overlay image is combined with a background image shown at the right part of FIG. 2 and is displayed on the screen.

In this way, a menu for a DVD is realized. A part of the navigation information is embedded in a stream using the NV_PCK for the purpose of realizing the menu without any problems even when an employed application is likely to have a difficulty in adjusting the synchronization timing, i.e.,

when menu information is to be dynamically updated in synchronization (sync) with the stream. One example of this is a case where the menu is displayed for five to ten minutes during playback of a movie.

FIG. 3 is an imaginary view of a VOB of a DVD. Video data, audio data, and subtitle data (FIG. 3A) are each divided into packets or packs based on the MPEG system (ISO/IEC13818-1) standard (FIG. 3B), and the packets or packs are multiplexed to generate one MPEG program stream (FIG. 3C). Here, the above-described NV_PCKs each including a button command for realizing interactivity are also multiplexed into the MPEG program stream.

Multiplexing employed in the MPEG system is characterized in that data packets to be multiplexed are each made up of a bit sequence based on its decoding order, but a bit sequence present between adjacent data packets to be multiplexed, i.e., between video, audio, and subtitle data, is not always based on its playback order, i.e., its decoding order. This happens because a decoder model of an MPEG system stream (generally referred to as a "System Target Decoder", or an "STD" (FIG. 3D)) has a decoder buffer corresponding to each elementary stream generated after demultiplexing, and so can temporarily accumulate demultiplexed data in the corresponding decoder buffer until its decoding timing. The size of each decoder buffer differs depending on the

corresponding elementary stream. A decoder buffer provided for video data has a capacity of 232kB, a decoder buffer provided for audio data has a capacity of 4kB, and a decoder buffer provided for subtitle data has a capacity of 52kB. Accordingly, the timing at which data is inputted into a decoder buffer differs depending on each elementary stream. Therefore, there is a difference between the order in which bit sequences of packets are arranged in an MPEG system stream and the order in which these bit sequences are decoded.

To be more specific, subtitle data multiplexed together with video data is not always decoded at the same timing at which the video data is decoded.

(Problems to be Solved by the Invention)

With recent advances in the IT technology, movie viewing is no longer restricted to a style where a viewer connects a DVD to a TV and plays back the DVD, but people can watch movies on PCs nowadays. As a result, the conventional division between DVDs as AV devices and PCs as non-AV devices has increasingly become vague. In the future, new types of entertainment produced by combining AV content recorded on DVDs with software programs such as a program language "Java", conventionally used in the PC world, or with the Internet are expected to be widespread.

As described in the "Related Art", an SD-DVD has such

a structure where AV data and commands for controlling its management information and scenarios are recorded on the disc. The DVD player interprets and executes the commands, to play back the AV data. This structure therefore is designed simply to enable the AV data to be played back. On the other hand, a program language such as Java does not involve the concept of synchronous playback on a time line as employed for the AV playback, and therefore fails to ensure playback of programs in synchronization with AV data. Further, in view of functions unique to AV devices such as fast-forward, rewind, and pause functions, a mechanism for synchronous playback of programs with AV data is yet to be established.

SUMMARY OF THE INVENTION

In view of the above problems, the present invention therefore aims at providing an HD-DVD technique for realizing a new world of applications where disc media are combined with computers, and with the internet, which has been impossible with the conventional DVD technique.

To solve the above problems, the invention relating to Claim 1 is a disc medium on which at least video data, audio data, management information for managing the video data and the audio data, and programs are recorded, characterized in that the programs include at least two programs that respectively operate in two different execution environments,

the disc medium is played by a playback apparatus that includes a decoder for playing back the video data the audio data, and a controller for selecting one program execution module from at least two different program execution modules corresponding to the at least two different execution environments, and the disc medium includes selection information to be used to select the one program execution module.

The invention relating to Claim 2 is the disc medium of Claim 1, characterized in that the management information includes at least stream management information and playback list information, the stream management information having attribute information of the video data and the audio data, the playback list information defining a playback sequence of the video data and the audio data, and the playback list information has selection information to be used to select one program execution module from the at least two different program execution modules.

The invention relating to Claim 3 is a playback apparatus that plays a disc medium on which at least video data, audio data, management information for managing the video data and the audio data, and programs are recorded, characterized in that the programs include at least two programs that respectively operate in two different execution environments, the disc medium includes selection information to be used

to identify an execution environment of each program, and the playback apparatus includes a decoder for playing back the video data the audio data, and a controller for selecting one program execution module from the at least two different program execution modules, based on the selection information.

The invention relating to Claim 4 is the playback apparatus of Claim 3, characterized in that the management information recorded on the disc medium includes at least stream management information and playback list information, the stream management information having attribute information of the video data and the audio data, the playback list information defining a playback sequence of the video data and the audio data, the playback list information has selection information to be used to select one program execution module from the at least two different program execution modules, and the controller selects the one program execution module based on the selection information.

The invention relating to Claim 5 is the playback apparatus of one of Claims 3 and 4, characterized by including a playback control module for receiving a playback pause request from a user, and transmitting the playback pause request to the decoder and the program execution module.

The invention relating to Claim 6 is a playback method for playing a disc medium on which at least video data, audio

data, management information for managing the video data and the audio data, and programs are recorded, characterized in that the programs include at least two programs that respectively operate in two different execution environments, and the disc medium includes selection information to be used to select an execution environment of each program, and the playback method including: a reading step of reading the management information; and a selecting step of selecting a program execution environment based on the selection information.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects, advantages and features of the invention will become apparent from the following description thereof taken in conjunction with the accompanying drawings that illustrate a specific embodiment of the invention.

In the drawings:

FIG. 1 shows the structure of a DVD;

FIG. 2 shows the structure of highlighting;

FIG. 3 shows one example of multiplexing in a DVD;

FIG. 4 shows a data hierarchy of an HD-DVD;

FIG. 5 shows a logical space provided in the HD-DVD;

FIG. 6 is a schematic block diagram of an HD-DVD player;

FIG. 7 is a block diagram showing the structure of the HD-DVD player;

FIG. 8 is a diagram for describing an application space provided in the HD-DVD;

FIG. 9 shows the structure of an MPEG stream (VOB);

FIG. 10 shows the structure of a pack;

FIG. 11 is a diagram for describing the relationship between an AV stream and the player structure;

FIG. 12 is a model diagram for continuously supplying AV data to a track buffer;

FIG. 13 shows a file structure of VOB information;

FIG. 14 is a diagram for describing a time map;

FIG. 15 is a diagram for describing a method for obtaining address information using the time map;

FIG. 16 shows the structure of a playlist file;

FIG. 17 shows the structure of a program file corresponding to the playlist;

FIG. 18 shows the structure of BD disc overall management information;

FIG. 19 shows the structure of a file in which a global event handler is stored;

FIG. 20 is a diagram for describing one example of a time event;

FIG. 21 is a diagram for describing one example of a user event;

FIG. 22 is a diagram for describing one example of a global even handler;

FIG. 23 shows the structure of a virtual machine;

FIG. 24 shows a player variable table;

FIG. 25 shows one example of an event handler (a time event);

FIG. 26 shows one example of an event handler (a user event);

FIG. 27 is a flowchart showing a basic process executed by the player;

FIG. 28 is a flowchart showing a playlist playback process;

FIG. 29 is a flowchart showing an event handling process;

FIG. 30 is a flowchart showing a subtitle handling process;

FIG. 31 is a diagram for describing a concept of BD applications;

FIG. 32 shows the logical structure of a BD in the second embodiment;

FIG. 33 is a diagram for describing a BD basic playback feature in the second embodiment;

FIG. 34 shows the directory/file structure of the BD in the second embodiment;

FIG. 35 is a block diagram showing the structure of a BD player in the second embodiment;

FIG. 36 shows the structure of a virtual machine of the BD player in the second embodiment;

FIG. 37 is a diagram for describing an example of execution in a Java class;

FIG. 38 is a flowchart showing a basic process executed by the BD player in the second embodiment;

FIG. 39 is a flowchart showing an event handling process in the second embodiment;

FIG. 40 is a flowchart showing switching between feature modules by the BD player;

FIG. 41 shows the structure of a playlist file in the third embodiment;

FIG. 42 is a conceptual diagram showing the relationship between a playlist and application attributes in the third embodiment;

FIG. 43 is a flowchart showing the operation of the player in the third embodiment;

FIG. 44 shows types of events in the third embodiment;

FIG. 45 shows event generation timings during playlist playback in the third embodiment;

FIG. 46 is a diagram for describing a pause starting process in the third embodiment; and

FIG. 47 is a diagram for describing a pause releasing process in the third embodiment.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

(First Embodiment)

The following describes a first embodiment of the present invention.

(Structure of Logical Data in Disc)

FIG. 4 shows the structure of an HD-DVD (hereafter sometimes referred to as a "BD"), in particular, the structure of a BD disc (104) that is a disc medium, and the structure of data (101, 102, and 103) stored in the disc. The BD disc (104) stores AV data (103), BD management information (102) including management information relating to the AV data and an AV playback sequence, and a BD playback program (101) for realizing interactivity. In the present embodiment, a BD disc is described focusing on its AV application for playing back AV contents such as movies, but the BD disc may of course be used as a recording medium for a computer such as a CD-ROM and a DVD-ROM.

FIG. 5 shows logical data stored in the above BD disc. Like other optical discs such as DVDs and CDs, the BD disc has a spiral recording area extending from an inner radius toward an outer radius of the disc, and has a logical address space for storing logical data between the lead-in at the inner radius and the lead-out at the outer radius. Also, a special area called a "BCA (Burst Cutting Area)" that can be accessed only using a drive is provided at the inner side of the lead-in. This area cannot be accessed from an application, and therefore, is often used by a copyright

protection technique or the like.

The logical address space stores, at its head, file system information (volume) that is followed by application data such as video data. As in the Description of the Related Art, the file system here intends to mean the UDF, the ISO9660, or the like. This file system enables logical data stored in the same manner as in a normal PC, to be read with use of the directory and file structure.

According to the directory and file structure of the BD disc in the present embodiment, the "BDVIDEO" directory is positioned immediately below the "ROOT" directory. In the "BDVIDEO" directory, data (101, 102, and 103 described above with reference to FIG. 4) such as AV contents and management information handled in an HD-DVD is stored.

Positioned below the "BDVIDEO" directory are the following seven types of files:

BD.INFO (file name being fixed);

This file is one type of "BD management information", and stores information about the overall BD disc. The file is the first to be read by the BD player.

BD.PROG (file name being fixed);

This file is one type of "BD playback program",

and stores a program relating to the overall BD disc.

XXX.PL ("XXX" being variable, and extension "PL" being fixed);

This file is one type of "BD management information", and stores playlist (PlayList) information containing a scenario. This file is provided for each playlist.

XXX.PROG ("XXX" being variable, and extension "PROG" being fixed);

This file is one type of "BD playback program", and stores a program provided for each playlist. A file body name of this file identifies the corresponding playlist ("XXX" matches).

YYY.VOB ("YYY" being variable, and extension "VOB" being fixed);

This file is one type of "AV data", and stores a VOB (the same as a VOB in the Description of the Related Art). This file is provided for each VOB.

YYY.VOBI ("YYY" being variable, and extension "VOBI"

being fixed);

This is one type of "BD management information", and stores management information relating to a VOB that is AV data. A file body name of this file identifies the corresponding VOB ("YYY" matches).

ZZZ.PNG ("ZZZ" being variable, and extension "PNG" being fixed);

This is one type of "AV data", and stores image data PNG for constituting a subtitle and a menu (an image format standardized by the W3C). This file is provided for each PNG image.

(Structure of the Player)

The following describes the structure of the player that plays the above BD disc, with reference to FIGS. 6 and 7.

FIG. 6 is a block diagram roughly showing the functional structure of the player.

Data stored in the BD disc (201) is read via an optical pickup (202). The read data is stored into a special memory according to a type of the read data. The BD playback program (data written in the file "BD.PROG" or the file "XXX.PROG") is stored into a program storage memory (203), the BD management information ("BD.INFO", "XXX.PL", or "YYY.VOBI") is stored into a management information storage memory (204), and the

AV data ("YYY.VOB" or "ZZZ.PNG") is stored into an AV storage memory (205).

The BD playback program stored in the program storage memory (203) is processed by a program processing unit (206). The BD management information stored in the management information storage memory (204) is processed by a management information processing unit (207). The AV data stored in the AV storage memory (205) is processed by a presentation processing unit (208).

The program processing unit (206) receives information about a playlist to be played back, event information such as the execution timing of a program, or the like, from the management information processing unit (207), and executes a program. Here, the program can dynamically change the playlist to be played back. To do so, an instruction to play back the playlist is sent to the management information processing unit (207). The program processing unit (206) receives an event from the user, i.e., a user request via the remote controller. When there is a program corresponding to the received user event, the program processing unit (206) executes the program.

The management information processing unit (207) receives an instruction from the program processing unit (206), and analyzes a playlist corresponding to the instruction and management information for a VOB corresponding to the playlist.

Then, the management information processing unit (207) instructs the presentation processing unit (208) to play back AV data corresponding to the VOB. Also, the management information processing unit (207) receives reference time information from the presentation processing unit (208), and instructs the presentation processing unit (208) to stop playback of the AV data based on the time information. Also, the management information processing unit (207) generates an event showing the program execution timing and sends the generated event to the program processing unit (206).

The presentation processing unit (208) has decoders in one-to-one correspondence with video data, audio data, and subtitle/image data. According to an instruction sent from the management information processing unit (207), the presentation processing unit (208) decodes and outputs AV data. Video data and subtitle/image data are drawn on special planes after being decoded. To be specific, the video data is drawn on a video plane (210), and the subtitle/image data is drawn on an image plane (209). A superimposing unit (211) executes a superimposing process of video, and then outputs the resulting data onto a display device such as a TV.

As shown in FIG. 6, the BD player has the structure based upon the structure of such data stored in the BD disc as shown in FIG. 4.

FIG. 7 is a block diagram showing in detail the structure

of the above-described player. In FIG. 7, the AV storage memory (205) corresponds to and is divided into an image memory (308) and a track buffer (309), the program processing unit (206) corresponds to and is divided into a program processor (302) and a UOP manager (303), the management information processing unit (207) corresponds to and is divided into a scenario processor (305) and a presentation controller (306), and the presentation processing unit (208) corresponds to and is divided into a clock (307), a demultiplexer (310), an image processor (311), a video processor (312), and an audio processor (313).

The VOB data (MPEG stream) read from the BD disc (201) is stored into the track buffer (309). The image data (PNG) read from the BD disc (201) is stored into the image memory (308). The demultiplexer (310) reads VOB data stored in the track buffer (309) based on the time shown by the clock (307), and sends video data to the video processor (312) and audio data to the audio processor (313). The video processor (312) and the audio processor (313) each are made up of a decoder buffer and a decoder as defined by the MPEG system standard. To be more specific, video data and audio data sent from the demultiplexer (310) respectively to the video processor (312) and to the audio processor (313) are temporarily stored in the respective decoder buffers, and are subjected to the decoding process by the respective decoders in accordance

with the clock (307).

The PNG stored in the image memory (308) can be processed by the following two methods.

In the case of image data for a subtitle, the presentation controller (306) sends an instruction about its decoding timing. The scenario processor (305) once receives time information sent from the clock (307), and instructs the presentation controller (306) to start and end displaying the subtitle at the subtitle display time (start and end time) to enable an appropriate display of the subtitle. The image processor (311) receives an instruction to decode and display the subtitle from the presentation controller (306), and so reads the corresponding PNG data from the image memory (308), decodes the read PNG data, and draws the resulting data on the image plane (314).

In the case of image data for a menu, the program processor (302) sends an instruction about its decoding timing. The timing at which the program processor (302) sends such an instruction for decoding cannot be generalized as it depends on a BD program that is being processed by the program processor (302).

As described above with reference to FIG. 6, the image data and the video data are respectively written to the image plane (314) and the video plane (315) after decoded, and then subjected to the superimposing process and outputted by a

superimposing unit (316).

The management information (scenario and AV management information) read from the BD disc (201) is stored into a management information storage memory (304), whereas the scenario information ("BD.INFO" and "XXX.PL") is read and processed by the scenario processor (305). Also, the AV management information ("YYY.VOBI") is read and processed by the presentation controller (306).

The scenario processor (305) analyzes information about a playlist, and sends an instruction about a VOB referred to by the playlist and a playback position of the VOB, to the presentation controller (306). The presentation controller (306) analyzes management information ("YYY.VOBI") for the VOB, and instructs a drive controller (317) to read the VOB.

The drive controller (317) follows the instruction sent from the presentation controller (306) and reads AV data corresponding to the VOB by moving the pickup. The read AV data is stored into the image memory (308) or into the track buffer (309) as described above.

Also, the scenario processor (305) monitors the time shown by the clock (307), and sends an event to the program processor (302) at the timing set by the management information.

The BD program ("BD.PROG" or "XXX.PROG") stored in the

program storage memory (301) is executed by the program processor 302. The program processor (302) executes the BD program when receiving an event sent from the scenario processor (305) or when receiving an event sent from the UOP manager (303). The UOP manager (303) generates an event and sends the generated event to the program processor (302) when receiving a request from the user via a key provided on the remote controller.

(Application Space)

FIG. 8 shows an application space provided in an HD-DVD.

In the application space of the HD-DVD, a playlist (PlayList) is a unit of playback. The playlist has a static scenario that is made up of a playback sequence of cells (Cells), and a dynamic scenario that is written using a program. Without the dynamic scenario written using the program, the playlist is played back simply by sequentially playing back the cells included therein. Here, the playback of the playlist is completed when all the cells included therein are played back. On the other hand, the program can designate playback beyond its playlist, and also can dynamically change a playback target according to the user's selection or according to the status of the player. A typical example of this is a menu. For an HD-DVD, a menu allows the user to dynamically select a scenario to be played back, i.e., a playlist.

A program here is an event handler to be executed by a time event or a user event.

A time event is generated based on time information embedded in a playlist. The time event corresponds to an event sent from the scenario processor (305) to the program processor (302) described above with reference to FIG. 7. When the time event is generated, the program processor (302) executes an event handler that is associated with the time event using an ID. As described above, a program to be executed can designate playback of another playlist. When the program designates playback of another playlist, playback of the present playlist is stopped, and playback of the designated other playlist is started.

A user event is generated by a user operation of a key on the remote controller. User events can be roughly categorized into two types. A user event of one type is an event for a menu selection and is generated by operating the cursor keys (the "up", "down", "left", and "right" arrow keys) or the "return" key. An event handler corresponding to a user event for a menu selection is only valid in a limited duration within a playlist (a valid duration of each event handler is set as information about a playlist). When one of the "up", "down", "left", and "right" arrow keys and the "return" key on the remote controller is pressed, a valid event handler is retrieved. When a valid event handler is found, the event

handler is executed. In the other cases, the user event for a menu selection is ignored.

A user event of the other type is an event for a menu call and is generated by operating the "menu" key . When a user event for a menu call is generated, a global event handler is called. A global event handler does not depend on a playlist, but is valid at all times. By using this function, a menu call in a DVD (calling an audio menu, a subtitle menu, or the like during playback of a title, changing an audio or a subtitle, and then resuming the playback of the title from the suspended position) can be realized.

A cell (Cell) that is a unit for constituting a static scenario in a playlist refers to some or all of playback sections of a VOB (an MPEG stream). A cell has information about playback sections of a VOB to which the cell refers to, as information about a start time and an end time of the playback sections within the VOB. VOB management information (VOBI) that makes a pair with each VOB internally has a time map (Time Map, or TM). By referring to the time map, a start address and an end address of reading data within the VOB (i.e., in the corresponding file "YYY. VOB") can be obtained using the above-described start and end time of playing the VOB. It should be noted here that a time map is described in detail later in this specification.

(Detailed Description of VOB)

FIG. 9 shows the structure of an MPEG stream (VOB) that is used in the present embodiment.

As shown in FIG. 9, a VOB is made up of a plurality of VOBUs (Video Object Units). Each VOBU is based on a GOP (Group Of Pictures) defined for an MPEG video stream, and is a multiplexed stream of one playback unit also including audio data. A VOBU has a duration of 0.4 to 1.0 sec., usually a playback duration of 0.5 sec. This value is derived from the structure of a GOP of an MPEG video stream usually having 15 frames/sec. (in the case of NTSC system).

A VOBU internally has video packs (V_PCK) and audio packs (A_PCK). Each pack has one sector, i.e., 2kB in the present embodiment.

FIG. 10 shows the structure of a pack.

As shown in FIG. 10, elementary data such as video data and audio data is continuously stored into a data storage area of a packet called a "payload". A payload is provided with a packet header. The payload and the packet header together form a packet. A packet header includes information indicating a type of a stream of data stored in its payload, i.e., a video stream or an audio stream. If there are a plurality of video streams or a plurality of audio streams, the packet header stores an ID (stream_id) for identifying which of these streams the data belongs to. The packet header

also includes time stamps "DTS" and/or "PTS" that respectively indicate decoding time information and display time information of the payload. The PTS and DTS are not always stored in each packet header. The MPEG defines the rule relating to storage of the PTS and DTS as written in detail in the MPEG system (ISO/IEC13818-1) standard, and so the rule is not described in this specification.

The packet is provided with another header (a pack header). The packet and the pack header together form a pack. In the pack header, a time stamp "SCR (System Clock Reference)" indicating the timing at which the pack is made through a demultiplexer and is inputted into the corresponding decoder buffer.

(Interleaved Recording of VOB)

The following describes interleaved recording of a VOB file, with reference to FIGS. 11 and 12.

The upper part of FIG. 11 is a part of the above-described player structure. As shown in the figure, data stored in the BD disc is inputted, via the optical pickup, into the track buffer when the data is a VOB, i.e., an MPEG stream, and into the image memory when the data is PNG, i.e., image data.

The track buffer employs FIFO (first-in first-out), and so input VOB data is sent to the demultiplexer in the order of input. Here, each pack is read from the track buffer

according to the above SCR, and data is sent to the video processor or the audio processor via the demultiplexer. On the other hand, in the case of image data, the presentation controller sends an instruction as to which image is to be drawn. Here, image data used for the drawing is deleted from the image memory if the image data is for a subtitle, and is left as it is in the image memory if the image data is for a menu. This is because drawing the menu partly depends on a user operation and so the same image may want be drawn a plural number of times.

The lower part of FIG. 11 shows interleaved recording of a VOB file and a PNG file on the BD disc. Typically, on a ROM, such as a CD-ROM and a DVD-ROM, AV data of a series of continuous playback sections is continuously recorded. For such continuously recorded data, a drive may simply read data and send the data to the player side sequentially. However, if such continuous data is divided and recorded as separate playback units being distributed on the disc, a seek operation is required between continuous playback sections. This seek operation interrupts reading of data, and further interrupts supplying of data. For an HD-DVD, too, it is preferable to record VOB files in a continuous recording area. However, there may be data such as subtitle data that is to be played back in synchronization with video data in a VOB. In this case, the subtitle data is required to be read from

the BD disc in some way in the same manner as a VOB file.

As one method for reading subtitle data, image data (PNG files) for subtitles may be read at once before a VOB is started to be played back. This method is unrealistic because a memory with a large capacity is required for temporarily storing such image data for subtitles.

In view of this, the present embodiment employs a method of dividing a VOB file into a plurality of blocks, and recording the VOB blocks and the image data blocks as being interleaved with one another. The lower part of FIG. 11 describes such interleaved recording.

By appropriately positioning the VOB file blocks and the image data blocks as being interleaved with one another, blocks of image data can be stored into the image memory at an appropriate timing without a temporal storage memory with a large capacity as described above. In this case, however, reading of VOB data is naturally suspended while image data is being read.

FIG. 12 is a diagram for describing a VOB data continuous supplying model employing a track buffer that solves the above-described problem.

As described above, data of a VOB is once accumulated in the track buffer. If the input rate of data into the track buffer is set different from the output rate of data from the track buffer, the amount of data accumulated in the track

buffer increases while data is continuously read from the BD disc.

Assume here that the input rate of data into the track buffer is "Va" and the output rate of data from the track buffer is "Vb". Assume also that a continuous recording area of the VOB extends from a logical address "a1" to a logical address "a2" as shown in the upper part of FIG. 12. Assume further that a section between the addresses "a2" and "a3" stores image data, and so VOB data cannot be read from this section.

The lower part of FIG. 12 shows the internal state of the track buffer. The horizontal axis indicates time, and the vertical axis indicates the amount of data accumulated in the track buffer. The time "t1" indicates the time at which reading of data at the address "a1" that is the start point of the continuous recording area of the VOB is started. Since this time, data is accumulated into the track buffer at such a rate that is a difference between the input rate "Va" and the output rate "Vb". The time "t2" indicates the time at which reading of data at the address "a2" that is the end point of the continuous recording area of the VOB is ended. To sum up, the amount of data in the track buffer increases at the rate "Va-Vb" from the time "t1" to the time "t2". The amount of accumulated data at the time "t2" can be obtained by the following expression.

Expression 1

$$B(t2)=(Va-Vb)*(t2-t1)$$

Subsequently, image data is continuously recorded until the address "a3" on the BD disc, and therefore, no data is inputted into the track buffer until the address "a3". This means that the amount of data accumulated in the track buffer decreases at the output rate of "-Vb". This phenomenon continues until the reading position "a3", i.e., the time "t3".

It is important to note here that there may be no VOB data to be supplied to the decoder and playback of the VOB may be stopped if the amount of data accumulated in the track buffer reaches zero before the time "t3". In other words, playback of the VOB can be continued as long as data remains in the track buffer at the time "t3".

This condition can be expressed using the following expression.

Expression 2

$$B(t2) \geqq -Vb*(t3-t2)$$

Accordingly, image data is to be positioned in such a manner that the expression 2 is satisfied.

(Structure of Navigation Information)

The following describes the structure of navigation information (BD management information) of an HD-DVD, with reference to FIGS. 13 to 19.

FIG. 13 shows the internal structure of a VOB management information file ("YYY.VOBI").

The VOB management information includes stream attribute information (Attribute) and a time map (TMAP) of the VOB. The stream attribute is made up of a video attribute (Video) and audio attributes (Audio#0 to Audio#m). In particular, a VOB can include a plurality of audio streams, and so the presence of data fields is indicated by the number of audio streams (Number) included in a VOB.

Fields for the video attribute (Video) and possible values for the fields are listed below.

Compression Format (Coding):

MPEG1

MPEG2

MPEG4

Resolution (Resolution):

1920x1080

1280x720

720x480

720x565

Aspect Ratio (Aspect):

4:3

16:9

Frame Rate (Framerate)

60

59.94

50

30

29.97

25

24

The following lists fields for an audio attribute (Audio) and possible values for the fields.

Compression Format (Coding):

AC3

MPEG1

MPEG2

LPCM

Channel Number (Ch.):

1 to 8

Language Attribute (Lang.):

The time map (TMAP) is a table storing information for each VOBU. The time map stores the number of VOBUs (Number) included in the VOB and VOBU information (VOBU#1 to VOBU#n) for each VOBU. The VOBU information for each VOBU includes a playback duration (Duration) of the VOBU and a data size (Size) of the VOBU.

FIG. 14 is a diagram for describing VOBU information in detail.

As is widely known, an MPEG stream has two aspects, namely, the time and the data size. For example, according to the AC3 - the compression specification for audio data - data is compressed at a fixed bit rate, and therefore, the relationship between time and an address can be obtained using a linear expression. However, in the case of MPEG video data, each frame has a fixed display duration. For example, one frame of the NTSC system has a display duration of 1/29.97 sec., but a data size of each frame after being compressed greatly varies depending on characteristics of an image or a type of a picture used for the compression. The picture type is one of I-picture, P-picture, and B-picture.

Accordingly, for MPEG video data, the relationship between time and an address cannot be expressed by a general expression.

Accordingly, the relationship between time and data for an MPEG system stream into which MPEG video data is multiplexed, i.e., a VOB, cannot be expressed by a general expression. Instead of a general expression, a time map (TMAP) is used to express the relationship between the time and the address within the VOB. As shown in FIG. 14, the time map (TMAP) is a table having as entries for each VOBU, the number of frames included in the VOBU and the number of packs included in the VOBU.

The following describes a method for using the time map (TMAP), with reference to FIG. 15.

When time information is given as shown in FIG. 15, a VOBU to which the time shown by the time information belongs is first retrieved. To be more specific, the number of frames included in each VOBU in the time map is summed up, and a VOBU positioned where the sum of the number of frames is equal to or exceeds the time shown by the time information (in terms of the number of frames) is retrieved as the VOBU to which the time shown by the time information belongs. The size of each VOBU immediately preceding the retrieved VOBU in the time map is summed up, and the resulting value is the start address of a pack to be read for playing back a frame including the time shown by the time information.

The following describes the internal structure of playlist information ("XXX.PL"), with reference to FIG. 16.

The playlist information is made up of a cell list (CellList) and an event list (EventList).

The cell list (CellList) is a playback cell sequence within the playlist. Cells are played back in the order written in the cell list. The cell list includes the number of cells (Number) and cell information (Cell#1 to Cell#n) for each cell.

The cell information (Cell#) has a VOB file name (VOBName), a valid section start time (In) and a valid section end time (Out) within the VOB, and a subtitle table (SubtitleTable). The valid section start time (In) and the valid section end time (Out) each are expressed by a frame number within the VOB. Using the above-described time map (TMAP), an address of VOB data that is required for playback can be obtained.

The subtitle table (SubtitleTable) stores subtitle data that is played back in synchronization with the VOB. Like audio, a subtitle can be provided in a plurality of languages. The first information stored in the subtitle table (SubtitleTable) is made up of the number of languages (Number) followed by tables for the languages (Language#1 to Language#k).

The table for each language (Language#) is made up of language information (Lang), the number of subtitle data

(Number) for subtitles to be displayed separately, and subtitle data (Speech#1 to Speech#j) for subtitles to be displayed separately. The subtitle data (Speech#) is made up of an image data file name (Name), a subtitle display start time (In), a subtitle display end time (Out), and a subtitle display position (Position).

The event list (EventList) is a table defining events given in the playlist. The event list is made up of the number of events (Number) and the events (Event#1 to Event#m). Each event (Event#) is made up of an event type (Type), an event ID (ID), an event generation time (Time), and a valid duration (Duration).

FIG. 17 shows an event handler table ("XXX.PROG") storing an event handler (a time event, and a user event for a menu selection) for each playlist.

The event handler table is made up of the number of defined event handler programs (Number), and the event handler programs (Program#1 to Program#n). Each event handler program (Program#) includes the definition of the start of an event handler (<event_handler> tag) and an event handler ID (ID) that makes a pair with the above-described event ID, followed by the program being written in parentheses "{" and "}" below a function.

The following describes the internal structure of information about the overall BD disc ("BD. INFO"), with

reference to FIG. 18.

The information about the overall BD disc is made up of a title list (TitleList) and an event table for a global event (EventTable).

The title list (TitleList) is made up of the number of titles (Number) within the disc, and title information (Title#1 to Title#n) for the titles. The title information (Title#) for each title includes a playlist table (PLTable) included in the title and a chapter list (ChapterList) within the title. The playlist table (PLTable) includes the number of playlists (Number) included in the title and names of the playlists (Name#1 to Name#m), i.e, a file name of each playlist.

The chapter list (ChapterList) includes the number of chapters (Number) included in the title and chapter information (Chapter#1 to Chapter#n) for the chapters. The chapter information (Chapter#) for each chapter has a cell table (CellTable) storing cells included in the chapter. The cell table (CellTable) is made up of the number of cells (Number) and entry information (CellEntry#1 to CellEntry#k) for the cells. The cell entry information (CellEntry#) for each cell includes a name of the playlist that includes the cell, and a cell number written in the playlist.

The event list (EventList) is made up of the number of global events (Number) and global event information (Event#2 to Event#m) for the global events. It should be noted here

that the global event that is defined first is called the "first event" (FirstEvent). The first event is an event called first when the BD disc is inserted into the player. The event information (Event#) for each global event simply includes an event type (Type) and an event ID (ID).

FIG. 19 shows a table for a global event handler program ("BD.PROG").

This table stores the same items as the event handler table described above with reference to FIG. 17.

(Mechanism for Generating Event)

The following describe a mechanism for generating an event, with reference to FIGS. 20 to 22.

FIG. 20 shows one example of a time event.

As described above, a time event is defined by the event list (EventList) in the playlist information ("XXX.PL"). To generate an event defined as a time event, i.e., an event whose event type (Type) is "TimeEvent", a time event whose ID is "Ex1" is sent from the scenario processor to the program processor when the present time reaches the event generation time ("t1"). The program processor retrieves an event handler having an event ID "Ex1", and executes the retrieved event handler. In the present embodiment, for example, such an operation for displaying two button images can be performed.

FIG. 21 shows one example of a user event for a menu

operation.

As described above, a user event for a menu operation is also defined by the event list (EventList) in the playlist information ("XXX.PL"). To generate an event defined as a user event, i.e., an event whose event type (Type) is "UserEvent", a user event is placed into a ready state when the present time reaches the event generation time ("t1"). Here, this event itself is yet to be generated, but is in a ready state for a duration shown by the valid period information (Duration).

As shown in FIG. 21, when the user presses one of the "up", "down", "left", and "right" arrow keys and the "return" key on the remote controller, a UOP event is first generated by the UOP manager and is sent to the program processor. The program processor sends the UOP event to the scenario processor. The scenario processor tries to retrieve an event that is valid at the time when receiving the UOP event. When retrieving such a valid user event, the scenario processor generates a user event, and sends the generated user event to the program processor. The program processor retrieves an event handler that has an event ID "Ev1", and executes the retrieved event handler. In the present embodiment, for example, playback of the playlist#2 is started.

The generated user event does not include information about the key pressed by the user. Information about the

selected key is sent to the program processor by a UOP event, and is stored into a register SPRM (8) in a virtual player. The event handler program can obtain a value of this register and can execute a branch process.

FIG. 22 shows one example of a global event.

As described above, a global event is defined by the event list (EventList) of information about the overall BD disc ("BD. INFO"). An event defined as a global event, i.e., an event whose event type (Type) is "GlobalEvent", is generated only when the user operates a key on the remote controller.

When the user presses the "menu" key, the UOP manager first generates a UOP event and sends the UOP event to the program processor. The program processor sends the UOP event to the scenario processor. The scenario processor generates a global event corresponding to the UOP event, and sends the generated global event to the program processor. The program processor tries to retrieve an event handler with an event ID "menu", and executes the retrieved event handler. In the present embodiment, for example, playback of the playlist#3 is started.

In the present embodiment, a single "menu" key is provided. However, a plurality of "menu" keys may be provided as for a DVD. In this case, an ID for each "menu" key is to be defined.

(Virtual Player Machine)

The following describes the functional structure of the program processor, with reference to FIG. 23.

The program processor is a processing module internally having a virtual player machine. The virtual player machine is a functional model defined as an HD-DVD, and so does not depend on mounting onto a different HD-DVD player. To be more specific, it is ensured that this virtual player machine can function in the same manner in any HD-DVD player.

The virtual player machine has two major features, namely, a programming function and a player variable (register). The programming function defines the following two features as HD-DVD unique functions based on JavaScript.

Link Function: stop the present playback and start playback at a designated playlist, cell, and time

Link (PL#, Cell#, time)

PL# : playlist name

Cell# : cell number

time : playback start time within cell

PNG Draw Function: draw designated PNG data on the image plane

Draw (File, X, Y)

File : PNG file name

X : X coordinate position

Y : Y coordinate position

Image Plane Clear Function: clear a designated area of the image plane

Clear (X, Y, W, H)

X : X coordinate position

Y : Y coordinate position

W : X direction width

H : Y direction width

The player variables include system parameters (SPRMs) indicating the status of the player and general parameters (GPRMs) used for general purposes.

FIG. 24 shows a list of system parameters (SPRMs).

SPRM (0) : Language code

SPRM (1) : Audio stream number

SPRM (2) : Subtitle stream number

SPRM (3) : Angle number

SPRM (4) : Title number

SPRM (5) : Chapter number

SPRM (6) : Program number

SPRM (7) : Cell number

SPRM (8) : Key name

SPRM (9) : Navigation timer

SPRM (10) : Current playback time

SPRM (11) : Mixing mode for Karaoke

SPRM (12) : Country information for parental management

SPRM (13) : Parental level

SPRM (14) : Player set value (video)

SPRM (15) : Player set value (audio)

SPRM (16) : Language code for audio stream

SPRM (17) : Language code for audio stream (extension)

SPRM (18) : Language code for subtitle stream

SPRM (19) : Language code for subtitle stream (extension)

SPRM (20) : Player region code

SPRM (21) : reserved

SPRM (22) : reserved

SPRM (23) : Player status

SPRM (24) : reserved

SPRM (25) : reserved

SPRM (26) : reserved

SPRM (27) : reserved

SPRM (28) : reserved

SPRM (29) : reserved

SPRM (30) : reserved

SPRM (31) : reserved

It should be noted here that although the present embodiment describes the case where a programming function for the virtual player is based on JavaScript, it may instead be other programming functions based on B-Shell and Perl Script used in UNIX OS or the like. In other words, the present invention should not be limited to JavaScript.

(Program Examples)

FIGS. 25 and 26 show examples of event handler programs.

FIG. 25 shows one example of a menu that has two selection buttons.

A program shown at left in FIG. 25 is executed using a time event at the start of a cell (PlayList#1. Cell#1). Here, the general parameter "GPRM(0)" is initially set at "1". The GPRM(0) is used to identify a selected button in the program. In the initial state, the button "1" positioned at left is being selected.

Following this, PNG is drawn using a function "Draw" for each of the button "1" and the button "2". The button "1" is to draw a PNG image "1black.png" with the coordinate (10, 200) as the start point (left end). The button "2" is to draw a PNG image "2white.png" with the coordinate (330, 200) as the start point (left end).

Also, at the end of the cell, a program shown at right

in FIG. 25 is executed using a time event. Here, a "Link" function is used to designate playback of the cell from the start again.

FIG. 26 shows one example of an event handler of a user event for a menu selection.

An event handler has a written program corresponding to each case where the "left" arrow key, the "right" arrow key, or the "return" key on the remote controller is pressed. When the user presses a key on the remote controller, a user event is generated as described above with reference to FIG. 21 and an event handler shown in FIG. 26 is activated. This event handler executes a branch process using a value of the GPRM(0) identifying the selected button and the SPRM(8) identifying the selected key on the remote controller.

Condition 1) The button "1" is selected and the "right" arrow key is selected.

The GPRM(0) is set again at "2", and the button "2" at right is placed into a selected state.

Images corresponding to the buttons "1" and "2" are rewritten.

Condition 2) The "return (OK)" key is selected and the button "1" is selected.

Playback of the playlist#2 is started.

Condition 3) The "return (OK)" key is selected and the button "2" is selected.

Playback of the playlist#3 is started.

The processing is executed as described above.

(Player Processing Flow)

The following describes a processing flow of the player, with reference to FIGS. 27 to 30.

FIG. 27 is a flowchart showing a basic process executed before AV playback.

The BD disc is inserted (S101), so that the HD-DVD player reads and analyses the BD.INFO file (S102), and reads the BD.PROG file (S103). The BD.INFO and the BD.PROG are both temporarily stored in the management information storage memory and are analyzed by the scenario processor.

Following this, the scenario processor generates the first event by refereeing to first event (FirstEvent) information within the BD.INFO file (S104). The program processor receives the generated first event, and executes an event handler corresponding to the received first event (S105).

The event handler corresponding to the first event is expected to store playlist information to be played back first. If playback of the playlist is not instructed, the player

simply waits for receiving a user event without playing back any data. In this case, the player waits for receiving a user event (S201). When the HD-DVD player receives a user operation of the remote controller, the UOP manager generates a UOP event and sends the UOP event to the program manager (S202).

The program manager judges whether the UOP event is caused by the "menu" key or not (S203). When judging that the UOP event is caused by the "menu" key, the program manager sends the UOP event to the scenario processor, so that the scenario processor generates a user event (S204). The program processor executes an event handler corresponding to the generated user event (S205).

FIG. 28 is a flowchart showing a process from the start of PL playback to the start of VOB playback.

As described above, playback of the playlist is started by the first event handler or the global event handler (S301). The scenario processor reads and analyses the playlist information "XXX.PL" as information necessary to play back the playlist (S302), and reads the program information "XXX.PROG" corresponding to the playlist (S303). Following this, the scenario processor starts playback of a cell based on cell information written in the playlist (S304). Playback of the cell indicates to send a request from the scenario processor to the presentation controller, and so the presentation controller starts AV playback (S305).

When the AV playback is started (S401), the presentation controller reads and analyzes an information file (YYY.VOBI) for a VOB corresponding to the cell to be played back (S402). The presentation controller identifies a VOBU to be played back and its address using the time map, and designates the reading address to the drive controller, so that the drive controller reads the corresponding VOB data (S403). The VOB data is sent to the decoder, and playback of the VOB data is started (S404).

The VOB playback is continued until the end of the playback sections of the VOB (S405). After the end of the playback sections, the processing advances to playback of the next cell in step S304. When no next cell is present, the playback is stopped (S406).

FIG. 29 is a flowchart showing a process after the start of the AV playback.

The HD-DVD player is an event-driven type player model. When playback of the playlist is started, event handling processes of a time event, a user event, and a subtitle display are activated one after another, and these event handling processes are executed in parallel.

The processing flow from steps S501 to S505 is an event handling process of a time event.

Playback of the playlist is started (S501). A judgment is performed as to whether the playback of the playlist is

completed or not (S502). Following this, the scenario processor judges whether the present time has reached the time event generation time or not (S503). When judging that the present time has reached the time event generation time, the scenario processor generates a time event (S504), and the program processor receives the time event and executes an event handler (S505).

When the judgment result shows that the present time has not reached the time event generation time in step S503, or when execution of the event handler is completed in step S505, the processing returns to step S502, and the above-described processing is repeated. Also, when the judgment result shows that the playlist playback is completed in step S502, the time event handling process is forcibly terminated.

The processing flow from steps S601 to S608 is an event handling process of a user event.

Playback of the playlist is started (S601). A judgment is performed as to whether the playback of the playlist is completed or not (S602). Following this, a judgment is performed as to whether a UOP has been received or not (S603). When the judgment result shows that a UOP has been received, the UOP manager generates a UOP event (S604), and the program processor receiving the UOP event judges whether the UOP event is a menu call or not (S605). When judging that the UOP event

is a menu call, the program processor instructs the scenario processor to generate an event (S607). The program processor executes an event handler (S608).

When the judgment result shows that the UOP event is not a menu call in step S605, the UOP event is an event caused by a cursor key or the "return" key. In this case, the scenario processor judges whether the present time is within the user event valid duration or not (S606). When judging that the present time is within the user event valid duration, the scenario processor generates a user event (S607), and the program processor executes an event handler (S608).

When the judgment result shows that a UOP has not been received in step S603, when the judgment result shows that the present time is not within the user event valid duration in step S606, or when execution of the event handler is completed in step S608, the processing returns to step S602, and the above-described processing is repeated. Also, when the judgment result shows that the playlist playback is completed in step S602, the user event handling process is forcibly terminated.

FIG. 30 shows a flowchart showing a subtitle handling process.

Playback of the playlist is started (S701). A judgment is performed as to whether the playback of the playlist is completed or not (S702). Following this, a judgment is

performed as to whether the present time has reached a subtitle draw start time (S703). When the judgment result shows that the present time has reached the subtitle draw start time, the scenario processor instructs the presentation controller to draw a subtitle, so that the presentation controller instructs the image processor to draw the subtitle (S704). When the judgment result shows that the present time has not reached the subtitle draw start time, the scenario processor judges whether the present time has reached a subtitle draw end time or not (S705). When the judgment result shows that the present time has reached the subtitle draw end time, the presentation controller instructs the image processor to erase the subtitle that is presently being drawn, so that the subtitle is erased from the image plane (S706).

When the subtitle draw step S704 is completed, when the subtitle erasing step S706 is completed, or when the judgment result shows that the present time has not reached the subtitle draw end time in the subtitle draw end time judging step S705, the processing returns to step S702, and the above-described processing is repeated. Also, when the judgment result shows that the playlist playback is completed in step S702, the subtitle handling process is forcibly terminated.

(Second Embodiment)

The following describes a second embodiment of the

present invention.

The second embodiment relates to a BD player that has a plurality of playback/execution systems. Because the second embodiment is fundamentally based on the first embodiment, the second embodiment is described focusing on its developments on or differences from the first embodiment.

FIG. 31 compares the world of BD applications with the world of DVD applications.

The DVD world is a closed world that is made up of a DVD disc and a DVD player. The DVD disc stores AV data and navigation information necessary for playback control of the AV data. The DVD player has a playback control program, and processes static data of the DVD disc using the playback control program.

On the other hand, the BD (HD-DVD) world is a world linked to the Internet so that new content or a new execution program can be downloaded and played/executed by a BD player (HD-DVD player), as can be seen from FIG. 31. The BD player has several playback modes. The user can enjoy video content such as movies, through a variety of viewing modes such as a DVD-compatible playback mode, a Browser mode, and a Java mode.

Java (TM) is middleware developed by Sun Microsystems. In recent years, Java has been widely used for consumer devices such as mobile phones in Japan and digital broadcasting DVB-MHP in Europe. Java is an object-oriented programming language

like C++ and the like. However, while C++ depends on the execution environment, i.e., the type of OS, Java defines a virtual machine (Java Virtual Machine) that is mounted on major OSs including Windows and Linux. Due to this, Java is receiving attention as an OS-independent execution system. Even when the execution environment differs depending on each manufacturer, as is likely to be the case for consumer devices, an application program can be executed regardless of the execution environment. For this reason, Java is employed in mobiles phones and STBs.

BDs (HD-DVDs) can also greatly benefit users by employing Java. For example, the same game can be played between BD players of different manufacturers.

FIG. 32 shows the structure of a BD in the second embodiment.

As shown in FIG. 32, other media such as an HDD, a memory card, and a network can be placed in the lowest layer in addition to a BD medium. This indicates that logical data of the BD can be recorded not only on the physical BD medium but also on various other media.

When compared with the structure in the first embodiment shown in FIG. 4, the remarkable characteristic of the BD in this embodiment is that the above-mentioned different playback modes are provided in the highest layer. This enables BD content to be played with various modes being switched, such

as the DVD-compatible playback mode (corresponding to the first embodiment), the Browser playback mode, and the Java playback mode.

A BD basic playback feature is a layer for realizing switch between different playback modes. This layer contains a playback control feature which is a basic feature necessary for playing the BD, and a player status.

FIG. 33 is a conceptual diagram of the BD basic playback feature.

As shown in FIG. 33, actual playback of a playlist is conducted by the BD basic playback feature, regardless of whether the playback mode is DVD-compatible or Java. Which is to say, the BD basic playback feature has an I/F for each playback mode such as the DVD-compatible playback mode. Each playback mode uses a corresponding I/F to instruct playback of a playlist or to acquire a language attribute.

Each playback mode, such as the DVD-compatible playback mode described in the first embodiment or the Java playback mode, offers programming functions for describing dynamic scenarios. Some programming functions such as a function for executing playback of a playlist are realized in the layer of the BD basic playback feature.

FIG. 34 shows the directory/file structure on the BD disc. A special directory BDVIDEO is provided immediately below a ROOT directory, and files necessary for playing the

BD are provided immediately below the BDVIDEO directory, as in the first embodiment.

The difference from the first embodiment lies in that a JCLASSES directory is provided immediately below the BDVIDEO directory as a subdirectory. This directory contains Java application classes used for playing the BD.

In FIG. 34, BD.CLASS is a Java class file. This class file is executed by a Java player program provided in the BD player. Other class files are called from the BD.CLASS.

FIG. 35 shows the structure of a player in the second embodiment. The following focuses on the difference from the first embodiment.

The difference from the BD player structure of the first embodiment shown in FIG. 7 lies in the structure of the program processor (302). In the BD player of the second embodiment, the program processor (302) includes a DVD-compatible module (3021), a Browser module (3022), a Java module (3023), a playback controller (3024), and a BD basic feature processor/BD-FF processor (3025).

The DVD-compatible module (3021), the Browser module (3022), and the Java module (3023) are switchable modules. These modules are managed and controlled by the playback controller (3024). When the Java class file BD.CLASS is provided on the BD disc, the playback controller (3024) selects the Java module (3023). The Java module (3023) reads the Java

class file BD.CLASS which is read from the BD disc and stored in the program storage memory (301), and executes it.

The BD-FF processor (3025) is a module for realizing the above-mentioned BD basic playback feature. To realize playback control, actually each of the DVD-compatible module (3021), the Browser module (3022), and the Java module (3023) calls the feature of the BD-FF processor (3025).

FIG. 36 shows the structure of a virtual player machine in the second embodiment.

As noted earlier, the virtual player machine of the second embodiment has a hierarchical structure. The BD-FF processor that provides the BD basic playback feature is placed in the lowest layer, and the DVD-compatible playback feature, the Browser playback feature, and the Java playback feature are placed in the highest layer.

The BD-FF processor has the following features necessary for control of BD playback, and parameters showing the player status.

Link feature: capable of designating the following parameters.

PL#: playlist name

Cell#: cell number

time: playback start time in the cell

PNG draw feature: capable of designating the following parameters.

File: PNG file name

X: X coordinate position

Y: Y coordinate position

Image plane clear feature: capable of designating the following parameters.

Clear(X, Y, W, H)

X: X coordinate position

Y: Y coordinate position

W: width in the X direction

H: width in the Y direction

32 system parameters (SPRMs) showing the status of the player.

SPRM(0) to SPRM(31): individual values are the same as those in the first embodiment.

16 general parameters (GPRMs) that can be used for general purposes.

GPRM(0) to GPRM()

In the case of the DVD-compatible playback feature, these features are expressed as the functions and variables of

JavaScript, and can be used by event handler programs as described in the first embodiment.

In the case of the Java playback feature, on the other hand, these features are expressed as BD special classes as shown in FIG. 36, and can be called from application programs. The BD special classes are the following.

1) Status class: BdStatus

The BdStatus class is automatically generated by a BD player object. The status of the player can be acquired or set using member functions of the BdStatus class.

The BdStatus class has the following member functions.

Language code acquisition / setting function

getLanguage()

setLanguage()

Audio stream number acquisition / setting function

getAudioStreamNumber()

setAudioStreamNumber()

Subtitle stream number acquisition / setting function

getSubtitleStreamNumber()

setSubtitleStreamNumber()

Angle number acquisition / setting function

getAngleNumber()

setAngleNumber()

Title number acquisition function

getTitleNumber()

Chapter number acquisition function

getChapterNumber()

Program number acquisition function

getProgramNumber()

Cell number acquisition function

getCellNumber()

Selected key information acquisition function

getExecutedKeyInfo()

Navigation timer acquisition function

getNavigationTimerValue()

Playback time information acquisition function

getCurrentElapseTime()

Karaoke mixing mode acquisition function

getKARAOKEMixingMode()

Parental country information acquisition function

getParentalCountryInfo()

Parental level acquisition function

getParentalLevel()

Player set value (video) acquisition function

getPlayerConfigurationForVideo()

Player set value (audio) acquisition function

getPlayerConfigurationForAudio()

Audio stream language code acquisition function

getLanguageForAudio()

Subtitle stream language code acquisition function

getLanguageForSubtitle()

Player region code acquisition function

getRegionCode()

Playback status acquisition function

getPlaybackStatus()

2) Playback class: BdPlayback

The BdPlayback class is automatically generated by the BD player object. The playback control of the player can be carried out using member functions of the BdPlayback class.

The BdPlayback class has the following member functions.

Playlist link function: start playback from a designated playlist, cell, and time.

linkPL (PL#, Cell#, time)

PL#: playlist name

Cell#: cell number

time: playback start time in the cell

PNG draw function: draw designated PNG data on the image plane.

drawImage (File, X, Y)

File: PNG file name

X: X coordinate position

Y: Y coordinate position

Image plane clear function: clear a designated area of the image plane

clearImage (X, Y, W, H)

X: X coordinate position

Y: Y coordinate position

W: width in the X direction

H: width in the Y direction

3) Event class: BdEvent

The BdEvent class is automatically generated by the BD player object. This class is generated at the timing of generating a user event and a time event described in the first embodiment, when the Java playback feature is executed. Which is to say, this corresponds to an object generated instead of a user event and a time event.

The BdEvent class has the following member function.

Setting function for event acquisition

setEventListener(this)

this: the object which calls the member function

FIG. 37 shows an example where the above Java class functions are actually used.

First, the player activates the BD player object. When BD.CLASS is present on the BD disc, the BD player object reads this class, and generates a BD object.

The BD object is not installed on the player side, but is an application program provided together with the BD disc. The BD object has an event handler as its member function. After the BD object is generated, setEventListener which is the member function of the event object is called to declare the availability of the event handler.

If a user event or time event described in the first embodiment is generated, the event object activates the event handler of the BD object set by the setEventListener function.

The event handler achieves a dynamic scenario, by executing functions such as getLanguage() which is a member function of the player status object and link() which is a member function of the playback object.

Processing flows of the BD player are described below, with reference to FIGS. 38 to 40.

Here, the following only describes the DVD-compatible feature and the Java feature, among the above-mentioned switchable features, i.e., the DVD-compatible feature, the Browser feature, and the Java feature. The Browser feature is certainly one of the switchable features. However, because the gist of the present invention is to switch between a plurality of features, the following only describe the two among the three features for simplicity.

FIG. 38 is a flowchart showing a basic process of the player after a disc is inserted.

When the BD disc is inserted (S1001), the BD player judges whether the BD.CLASS file is present (S1002). If the BD.CLASS file is not present, the processing advances to step S102 in the first embodiment, and the BD player performs the same processing as in the first embodiment.

If the BD.CLASS file is present, a BD object that is an instance of BD.CLASS is generated by the Java module (S1003). The BD object calls the member function of the event object, i.e., setEventListener, and declares the event handler (S1004).

Following this, the BD player reads BD.INFO (S1005), and generates the first event, i.e., FirstEvent (S1006). As a result, the event handler of the BD object is executed by the event object (S1007).

FIG. 39 is a flowchart showing a time event handling process after playback of a playlist starts.

After playback of a playlist starts (S1101), the player judges whether the playback of the playlist has been completed (S1102), and then judges whether time event execution time is reached (S1103). If so, the player generates a time event (S1104). As a result, the event handler of the BD object is executed by the event object (S1105).

If the time event time is not reached in step S1103 or after the event handler has been executed, the processing returns to step S1102 to judge the completion of the playlist

playback.

FIG. 40 is a flowchart showing a mode switching process of the BD player.

After the playlist playback starts (S1201), the player receives a request for switching the current mode (S1202). Step S1202 is repeated until a mode switching request is received from the user. Upon receiving a mode switching request, the player proceeds to a different step according to the current mode (S1203). If the current mode is the DVD-compatible mode, the playback controller switches the valid module from the DVD-compatible module to the Java module (S1204). If the current mode is the Java mode, the playback controller switches the valid module from the Java module to the DVD-compatible module (S1205).

Here, mode switching is carried out not by completely stopping an individual feature, but by sending an event sent from the scenario processor to one module so as to invalidate the dynamic scenario processing feature of the other module. With reference to FIG. 35 showing the structure, this switching is achieved by the playback controller selecting one of the DVD-compatible module, the Browser module, and the Java module and controlling the flow of an event.

(Third Embodiment)

The following describes a third embodiment of the present

invention.

The third embodiment relates to a BD player that has a plurality of playback/execution systems, as the second embodiment. Because the second embodiment is fundamentally based on the first and second embodiments, the third embodiment is described focusing on its developments on or differences from the first and second embodiments.

The third embodiment greatly differs from the second embodiment in that the player automatically performs switching between modes according to an intension of a title manufacture, unlike in the second embodiment where the mode switching in the BD player is performed by a user selection.

FIG. 41 shows the structure of a playlist file in the present embodiment.

The structure of the playlist file in the present embodiment is substantially the same as the structure described with reference to FIG. 16, except that an application flag (AppFlg) that indicates an application attribute of the playlist is additionally provided.

FIG. 42 is a conceptual diagram showing the relationship between a playlist and application attributes in the present embodiment.

As shown in FIG. 42, playlists 1, 2, and 3 are executed in the DVD-compatible mode (movie mode), playlists 4 and 5 are executed in the Java mode, and playlists 6 and 7 are executed

in the Browser mode. Which is to say, each playlist has its application attribute.

An application attribute of each playlist is identified by the above-mentioned application flag (AppFlg). An application flag has one of the following values.

00 DVD-compatible mode (movie mode)

01 Java mode

02 Browser mode

The following describes an operation flow of the player in the present embodiment, with reference to FIG. 43.

When a disc is inserted in the player (S1301), the player judges whether the BD.CLASS file is present on the disc (S1302). If the BD.CLASS file is present, the player generates a BD object as in the second embodiment (S1303). Then, the processing advances to step S1304. If the BD.CLASS file is not present, the processing advances directly to step S1304, and the player activates in the same manner as in the first embodiment (S101 and subsequent steps).

The subsequent processing flow until the start of playback of a playlist is the same as the processing flow described in the first embodiment. The processing flow at the time of activating a playlist is different from the processing flow in the first and second embodiments as

described below.

Before starting to play back a playlist (S1401), the player first checks an application attribute of the playlist (S1402). The application attribute is identified by an application flag (AppFlg) described above.

When the application attribute is "movie (DVD-compatible)", the player starts to play back the playlist in the movie mode (DVD-compatible mode) (S1403: based on the processing flow in the first embodiment). When the application attribute is "Java", the player starts to play back the playlist in the Java mode (S1404: based on the processing flow in the second embodiment). When the application attribute is "Browser", the player starts to play back the playlist in the Browser mode (S1405).

In this way, to play back each playlist, the player performs switching between playback modes, based on application attribute information of each playlist. The switching between playback modes is executed by the playback controller (3024) selecting one of (i.e., switching between) the DVD-compatible module (3021), the Browser module (3022), and the Java module (3023) as described with reference to FIG. 35.

FIG. 44 shows types of events used in the present embodiment.

As described in the first and second embodiments, the

player of the present invention instructs processing modules, the DVD-compatible module (3021), the Browser module (3022), and the java module (3023), to execute an event. The actual processing here is a program to be executed by each module.

Events used in the present embodiment include a playlist starting event (PLStart) generated when playback of a playlist is to be started, a playlist ending event (PLEnd) generated when playback of a playlist is to be ended, a time event (Time) generated at a predetermined timing, a pause starting event (PauseOn) generated when an instruction to pause playback is given by the user, a pause ending event (PauseOff) generated when an instruction to release a playback pause is given by the user, a special playback starting event (TPStart) generated when special playback is to be started, and a special playback ending event (TPEnd) generated when special playback is to be ended.

FIG. 45 shows event generation timings between the start and end of the playlist playback.

As shown in FIG. 45, when playback of a playlist is started, a playlist starting event (PLStart) is first generated. An application processing module that has received this event (ID=PLStart) executes an event handler corresponding to this event, so that playback of AV data is started.

During playback of the AV data, a time event (ID=Time) is generated at a timing marked in advance. The important

characteristic here is that playback of the AV data continues regardless of an event being generated. It should be noted here that a plurality of time events can be included in the playlist.

When playback of the AV data in the playlist is completed, a playlist ending event (PLEnd) is generated. An application processing module that has received this event (ID=PLEnd) executes an event handler corresponding to this event, so that the playback of the playlist is ended.

The events described with reference to FIG. 45 are all generated at predetermined timings. FIGS. 46 and 47 describe an event generation procedure of the case where a user request (UOP) to pause playback or to release a playback pause is given.

FIGS. 46 and 47 show a part of the player structure that is described above with reference to FIG. 35.

As shown in FIG. 46, when a request to execute a pause is given from the user, the UOP manager (303) generates UOP:PauseOn, and transmits the PauseOn to the BD-FF processor (3025). The BD-FF processor (3025) executes the PauseOn by way of two processes. The first process is to issue a request to the presentation controller (306) to suspend the operation of the decoder (Function Call: PauseOn). The second process is to generate a pause starting event (ID=PauseOn) and transmits the generated pause starting event to the Java module

(3023).

The Java module calls an event handler corresponding to the event, and executes the event handler. The event handler can describe a program to suspend its processing (wait) in a Java application that is currently being executed. Which is to say, with the use of the event handler, the operation of the Java application that is executed in synchronization with playback of AV data can be suspended.

FIG. 47 is a diagram for describing a pause releasing process. When a request to release a playback pause is given from the user, the UOP manager (303) generates UOP:PauseOff, and transmits the PauseOff to the BD-FF processor (3025). The BD-FF processor (3025) executes the PauseOff by way of two processes. The first process is to issue a request to the presentation controller (306) to resume the operation of the decoder (Function Call: PauseOff). The second process is to generate a pause ending event (ID=PauseOff) and transmits the generated pause ending event to the Java module (3023).

The Java module calls an event handler corresponding to the event, and executes the event handler. The event handler can describe a program to resume its processing in a Java application whose processing has been suspended. When combined with the use of a pause starting event, this event handler enables the Java program to be executed in synchronization with playback of AV data.

Here, although the above describes the case where the BD-FF processor (3025) generates a pause starting event (ID=PauseOn) or a pause ending event (ID=PauseOff) and transmits the event to the Java module (3023), the BD-FF processor transmits the event to the Browser module (3022) when an application attribute of the playlist is the Browser mode, and to the DVD module (3021) when an application attribute of the playlist is the DVD-compatible mode (movie mode).

Although the present embodiment describes the case where an application attribute is provided as being included in playlist information, an application attribute may be internal information of title information (Title#) for a title list (TitleList) of the BD disc management information file (BD. INFO). Also, an application attribute may not be provided for each playlist, but may be provided for each title that is made up of a plurality of playlists. In this case, the same application attribute may be provided for playlists included in one title. Alternatively, playlists may be managed not by titles but by other units, and playlists belonging to each unit may be given the same application attribute.

Moreover, although the present embodiment describes the case where data is recorded on a BD disc, a part of the data may not be recorded on the BD disc but may be recorded on other media such as a hard disk drive and a memory card.

Further, although the present embodiment describes the case where a DVD-compatible playback program is recorded in a file as being separate from a stream, the DVD-compatible playback program may be recorded as being embedded in a stream as in the case of a DVD.

(Other Modification Examples)

Although the present invention is described based on the above preferred embodiments, the present invention should not be limited to such. The following examples are also within the scope of the present invention.

(1) The apparatuses described in the above embodiments are specifically realized by a computer system that is made up of a microprocessor, a ROM, a RAM, a hard disk unit, a display unit, a keyboard, a mouse, and the like. The RAM or the hard disk unit stores a computer program. The functions of the apparatuses are realized by the microprocessor operating in accordance with the computer program.

(2) The present invention may be the methods described in the above embodiments. Also, the present invention may be a computer program realizing these methods on a computer, or digital signals representing the computer program.

Also, the present invention may be a computer-readable recording medium on which the computer program or the digital signals are recorded. Examples of the computer-readable

recording medium include a flexible disk, a hard disk, a CD-ROM, an MO, a DVD, a DVD-ROM, a DVD-RAM, a BD (Blu-ray Disc), and a semiconductor memory. Also, the present invention may be the computer program or the digital signals recorded on such a recording medium.

Also, the computer program or the digital signals may be transmitted via an electric communication line, a wireless/cable communication line, or a network such as the Internet.

Also, the present invention may be a computer system that is made up of a microprocessor and a memory. Here, the memory stores the computer program and the microprocessor operates in accordance with the computer program.

Also, the program or the digital signals may be executed on an independent computer system by transferring them as being recorded on the recording medium, or via the network and the like.

(3) The above embodiments and the above modification examples may be combined with one another.

(Effects of the Invention)

According to the present invention, a DVD program structure, Java, and a browser are realized by separate program execution modules independent from a basic playback feature unit (BD-FF processor). The basic player feature unit is for

managing AV playback and the status of the player. A controller (switch) selects a program execution module according to application attribute information held by each playlist. Based on the AV playback status, the basic playback feature unit generates a synchronization event and transmits the event to the selected program execution module. Also, when a special playback process such as a pause of AV playback is executed, the basic playback feature unit generates an event showing the special playback status, and transmits the event to the program execution module. An event handler program of the program execution module receives this event, and executes synchronous processing suited to the AV playback status such as a program pause. With the above structure and operation, the BD player can realize various applications including such a computer-based program as Java, and a browser, while maintaining synchronization with AV processing.

(Explanation of Reference Numerals)

201 BD disc

202 optical pickup

203 program storage memory

204 management information storage memory

205 AV storage memory

206 program processing unit

207 management information processing unit

208 presentation processing unit

209 image plane

210 video plane

211 superimposing unit

301 program storage memory

302 program processor

303 UOP manager

304 management information storage memory

305 scenario processor

306 presentation controller

307 clock

308 image memory

309 track buffer

310 demultiplexer

311 image processor

312 video processor

313 audio processor

314 image plane

315 video plane

316 superimposing unit

317 drive controller

3021 DVD-compatible module

3022 Browser module

3023 Java module

3024 playback controller

3025 BD-FF processor

S101 disc inserting step

S102 BD.INFO reading step

S103 BD.PROG reading step

S104 first event generating step

S105 event handler executing step

S201 UOP receiving step

S202 UOP event generating step

S203 menu call judging step

S204 event generating step

S205 event handler executing step

S301 playlist playback starting step

S302 playlist information (XXX.PL) reading step

S303 playlist program (XXX.PROG) reading step

S304 cell playback starting step

S305 AV playback starting step

S401 AV playback starting step

S402 VOB information (YYY.VOBI) reading step

S403 VOB (YYY.VOB) reading step

S404 VOB playback starting step

S405 VOB playback ending step

S406 next cell presence judging step

S501 playlist playback starting step

S502 playlist playback end judging step

S503 time event time judging step

S504 event generating step

S505 event handler executing step

S601 playlist playback starting step

S602 playlist playback end judging step

S603 UOP reception judging step

S604 UOP event generating step

S605 menu call judging step

S606 user event valid duration judging step

S607 event generating step

S608 event handler executing step

S701 playlist playback starting step

S702 playlist playback end judging step

S703 subtitle draw start judging step

S704 subtitle drawing step

S705 subtitle draw end judging step

S706 subtitle erasing step

S1001 disc inserting step

S1020 BD.CLASS detect judging step

S1003 BD object generating step

S1004 event handler declaring step

S1005 BD.INFO reading step

S1006 first event generating step

S1007 event handler processing step

S1101 playlist playback starting step

S1102 playlist end judging step

S1103 time event time judging step

S1104 time event generating step

S1105 event handler processing step

S1201 playlist playback starting step

S1202 mode switch judging step

S1203 present valid mode judging step

S1204 step for switching to Java mode

S1205 step for switching to DVD-compatible mode

S1301 disc inserting step

S1302 BD. CLASS presence judging step

S1303 BD object generating step

S1401 playlist playback starting step

S1402 application attribute judging step

S1403 DVD-compatible mode activating step

S1404 Java mode activating step

S1405 Browser mode activating step

Although the present invention has been fully described by way of examples with reference to the accompanying drawings, it is to be noted that various changes and modifications will be apparent to those skilled in the art. Therefore, unless such changes and modifications depart from the scope of the present invention, they should be construed as being included therein.

What is claimed is

1. A disc medium on which at least video data, audio data, management information for managing the video data and the audio data, and programs are recorded, characterized in that

the programs include at least two programs that respectively operate in two different execution environments,

the disc medium is played by a playback apparatus that includes a decoder for playing back the video data the audio data, and a controller for selecting one program execution module from at least two different program execution modules corresponding to the at least two different execution environments, and

the disc medium includes selection information to be used to select the one program execution module.

2. The disc medium of Claim 1, characterized in that

the management information includes at least stream management information and playback list information, the stream management information having attribute information of the video data and the audio data, the playback list information defining a playback sequence of the video data and the audio data, and

the playback list information has selection information to be used to select one program execution module from the at least two different program execution modules.

3. A playback apparatus that plays a disc medium on which at least video data, audio data, management information for managing the video data and the audio data, and programs are recorded, characterized in that

the programs include at least two programs that respectively operate in two different execution environments,

the disc medium includes selection information to be used to identify an execution environment of each program, and

the playback apparatus includes a decoder for playing back the video data the audio data, and a controller for selecting one program execution module from the at least two different program execution modules, based on the selection information.

4. The playback apparatus of Claim 3, characterized in that

the management information recorded on the disc medium includes at least stream management information and playback list information, the stream management information having

attribute information of the video data and the audio data, the playback list information defining a playback sequence of the video data and the audio data,

the playback list information has selection information to be used to select one program execution module from the at least two different program execution modules, and

the controller selects the one program execution module based on the selection information.

5. The playback apparatus of one of Claims 3 and 4, characterized by including

a playback control module for receiving a playback pause request from a user, and transmitting the playback pause request to the decoder and the program execution module.

6. A playback method for playing a disc medium on which at least video data, audio data, management information for managing the video data and the audio data, and programs are recorded, characterized in that

the programs include at least two programs that respectively operate in two different execution environments, and

the disc medium includes selection information to be used to select an execution environment of each program, and

the playback method including:

a reading step of reading the management information; and

a selecting step of selecting a program execution environment based on the selection information.

ABSTRACT OF THE DISCLOSURE

[Problem]

Although a conventional DVD (SD-DVD) has a scenario description structure and a program structure for realizing AV playback, it does not have such a structure that allows realization of a game as a computer program language including Java, or connection to the Internet. On the other hand, such a computer-based program execution system as Java or a browser does not have a mechanism for ensuring real-time feature or synchronous playback. A BD (HD-DVD) therefore fails to achieve its goal, the fusion of AV and PC/Internet. A mechanism is therefore required for realizing synchronous processing between a unit of processing execution of such a computer language as Java or a browser, and AV playback.

[Means of Solving the Problem]

A BD player of the present invention has a structure in which a DVD program structure, Java, and a browser are realized by separate program execution modules independent from a basic playback feature unit (BD-FF processor). The basic player feature unit is for managing AV playback and the status of the player. In the structure, a controller (switch) selects a program execution module according to application attribute information held by each scenario (playlist). Based on the AV playback status, the basic playback feature unit generates a synchronization event and transmits the event to the selected

program execution module. Also, when a special playback process such as a pause of AV playback is executed, the basic playback feature unit generates an event showing the special playback status, and transmits the event to the program execution module. An event handler program of the program execution module receives this event, and executes synchronous processing suited to the AV playback status such as a program pause.

[Effects]

The above means enables the BD player to realize various applications including such a computer-based program as Java, and a browser, while maintaining synchronization with AV processing.

[Selected Drawing]

FIG. 41

FIG.1

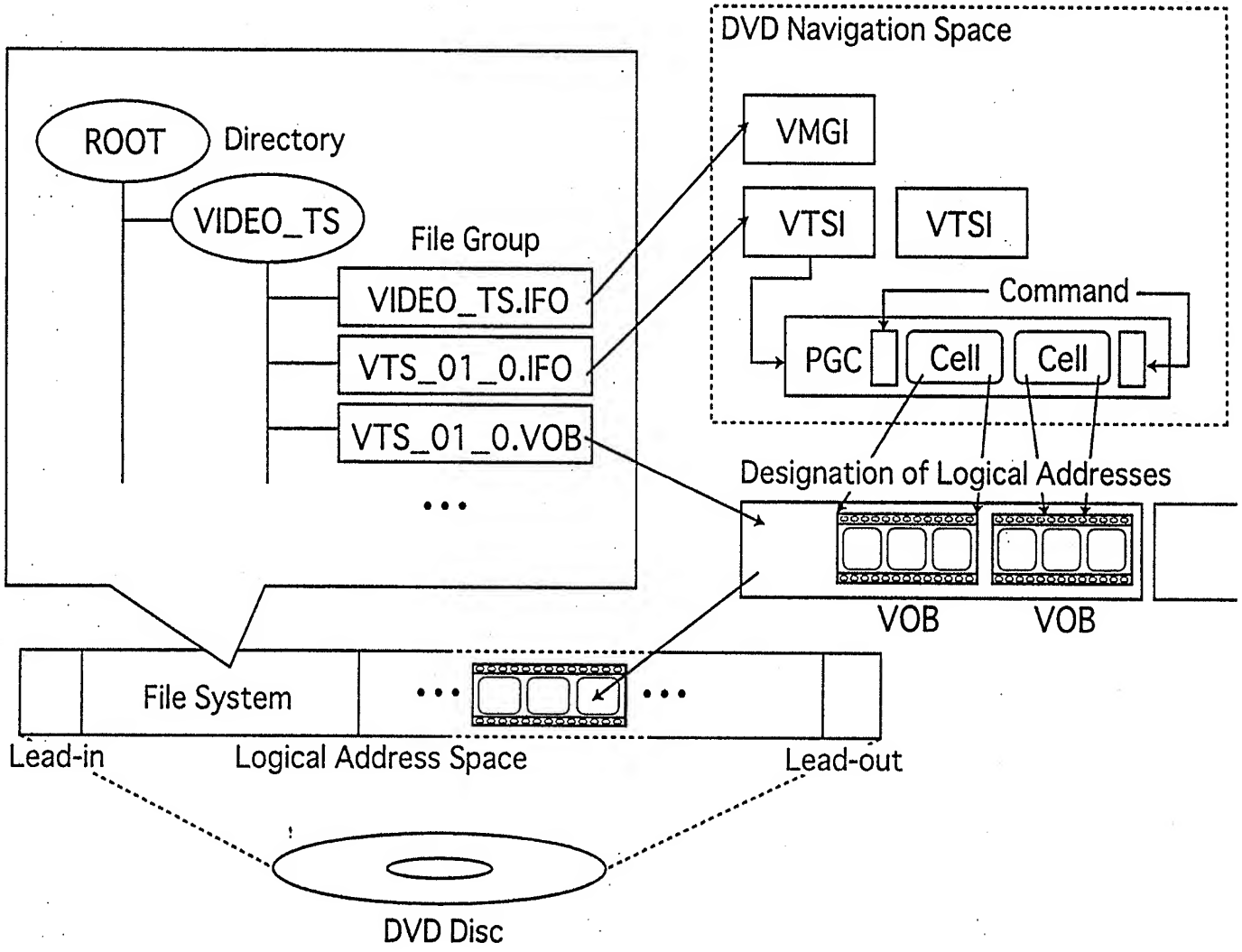

DVD Navigation Space
ROOT
Directory
VIDEO_TS
File Group
VIDEO_TS.IFO
VTS_01_0.IFO
VTS_01_0.VOB
VMGI
VTSI
VTSI
Command
PGC
Cell
Cell
Designation of Logical Addresses
VOB
VOB
File System
Lead-in
Logical Address Space
Lead-out
DVD Disc

FIG.2

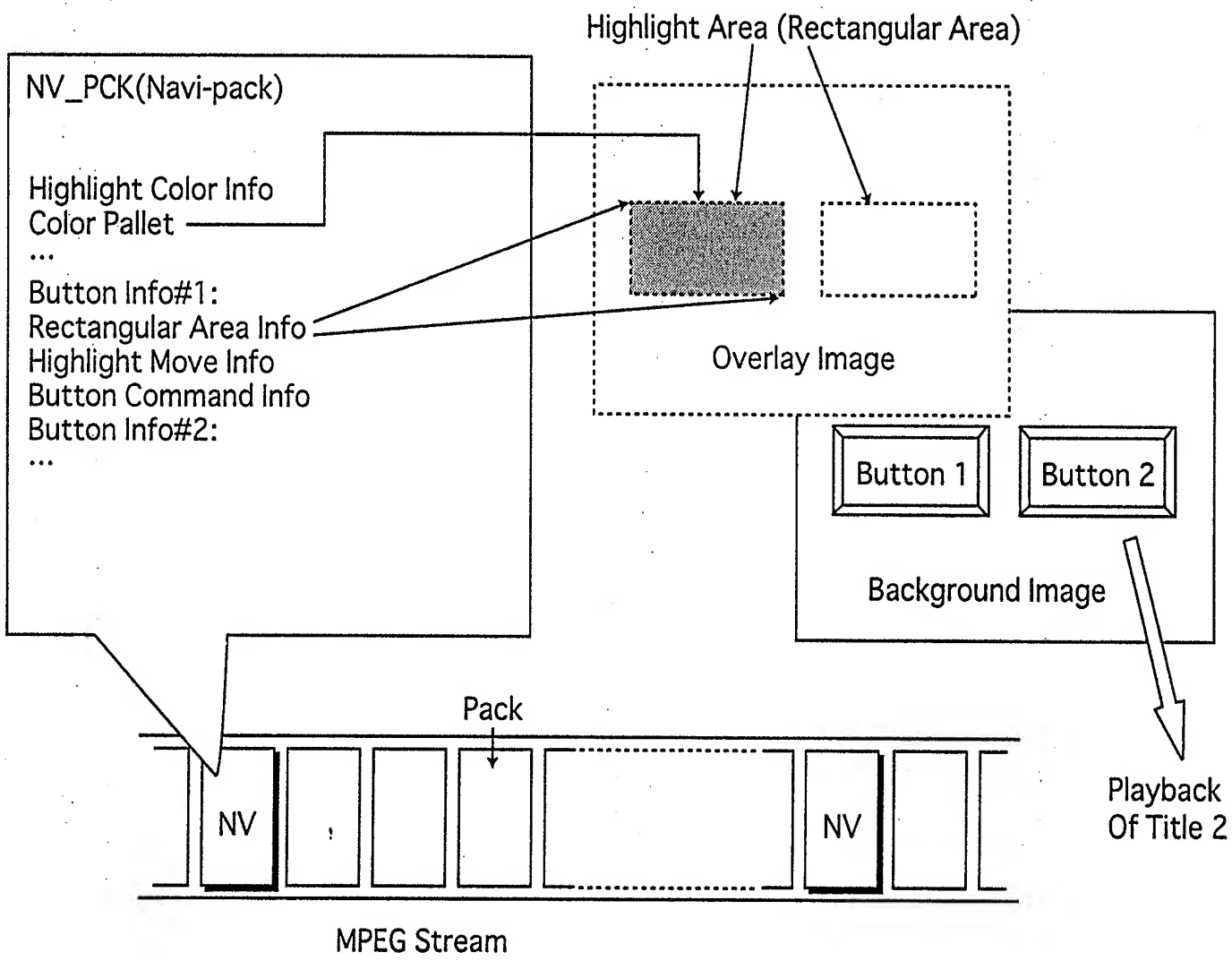

Highlight Area (Rectangular Area)
NV_PCK(Navi-pack)
Highlight Color Info
Color Pallet
...
Button Info#1:
Rectangular Area Info
Highlight Move Info
Button Command Info
Button Info#2:
...
Overlay Image
Button 1
Button 2
Background Image
Pack
NV
NV
Playback
Of Title 2
MPEG Stream

FIG.3

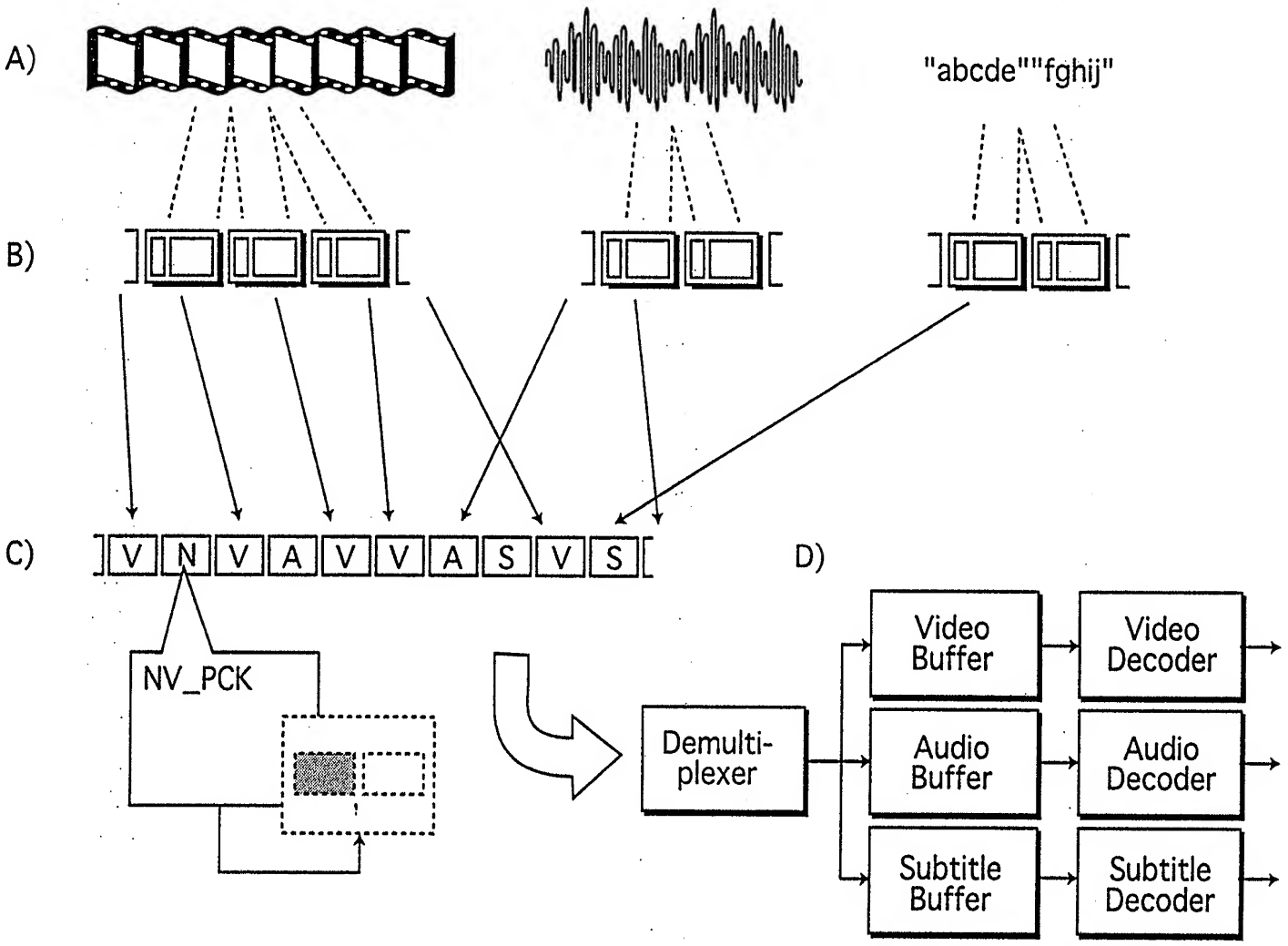

A)
"abcde""fghij"
B)
C)
V N V A V V A S V S
NV_PCK
D)
Demulti-plexer
Video Buffer
Audio Buffer
Subtitle Buffer
Video Decoder
Audio Decoder
Subtitle Decoder

FIG.4

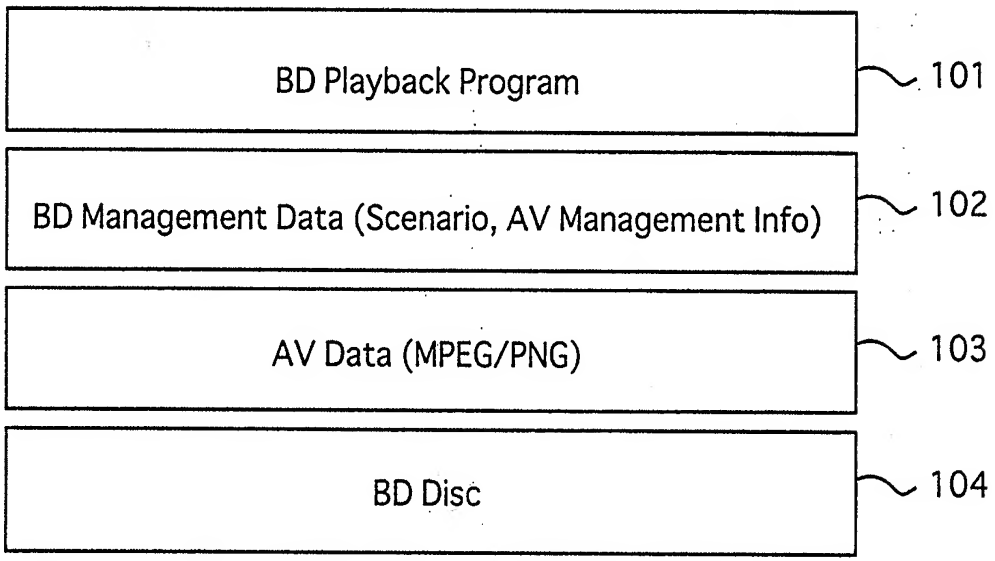

BD Playback Program
101
BD Management Data (Scenario, AV Management Info)
102
AV Data (MPEG/PNG)
103
BD Disc
104

FIG.5

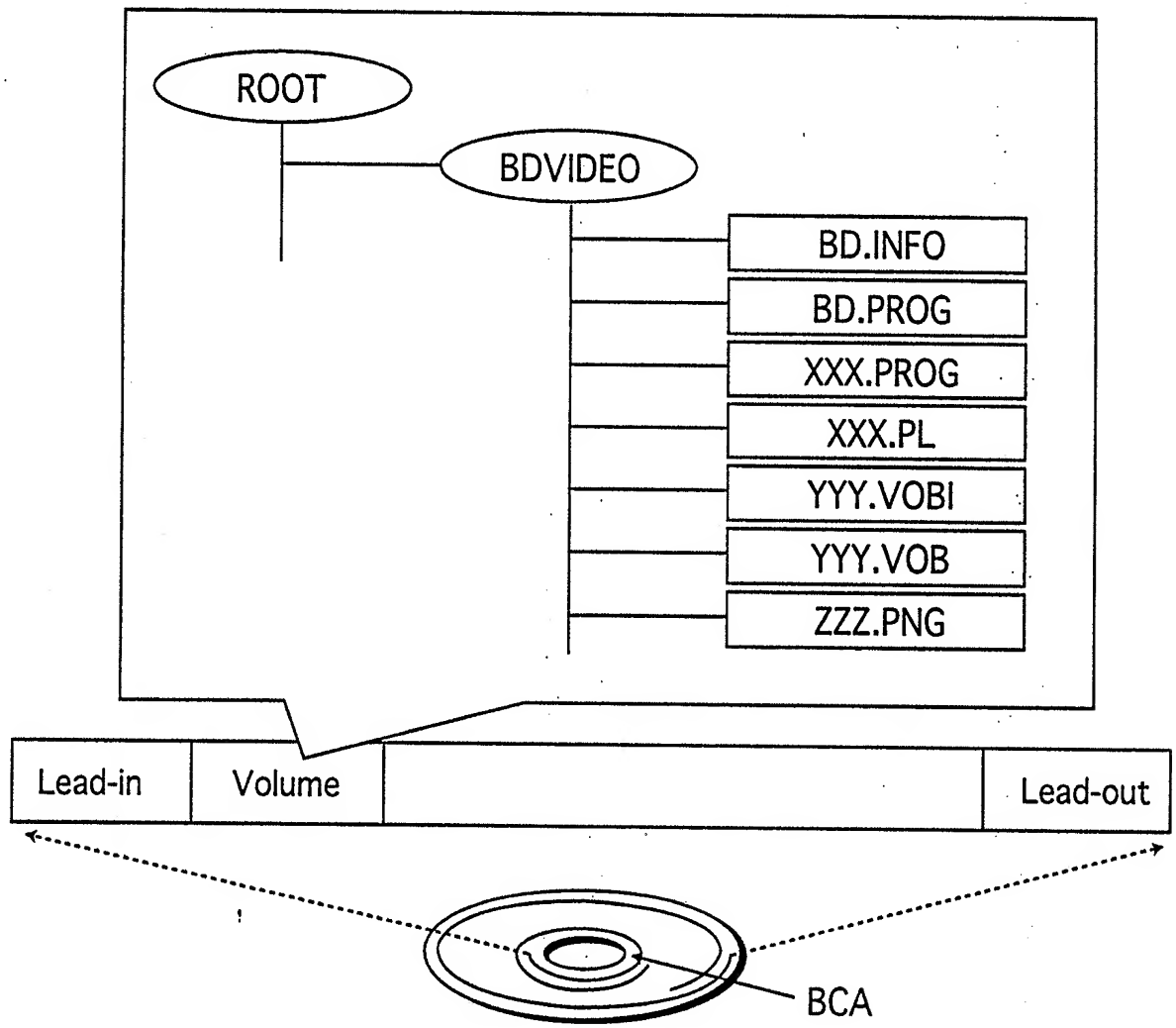

ROOT
BDVIDEO
BD.INFO
BD.PROG
XXX.PROG
XXX.PL
YYY.VOBI
YYY.VOB
ZZZ.PNG
Lead-in
Volume
Lead-out
BCA

FIG.6

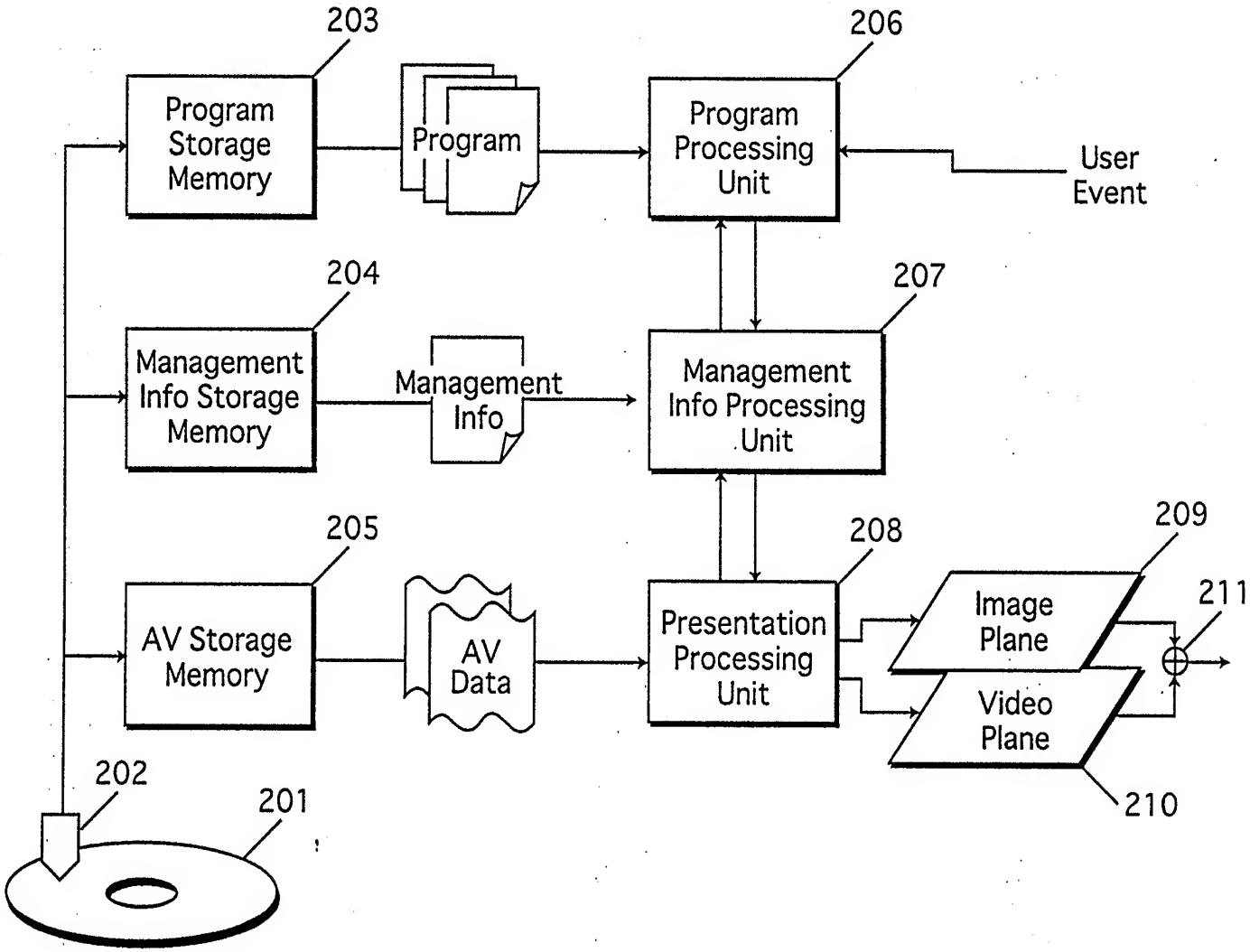

203
206
Program Storage Memory
Program
Program Processing Unit
User Event
204
207
Management Info Storage Memory
Management Info
Management Info Processing Unit
205
208
209
211
AV Storage Memory
AV Data
Presentation Processing Unit
Image Plane
Video Plane
202
201
210

FIG.7

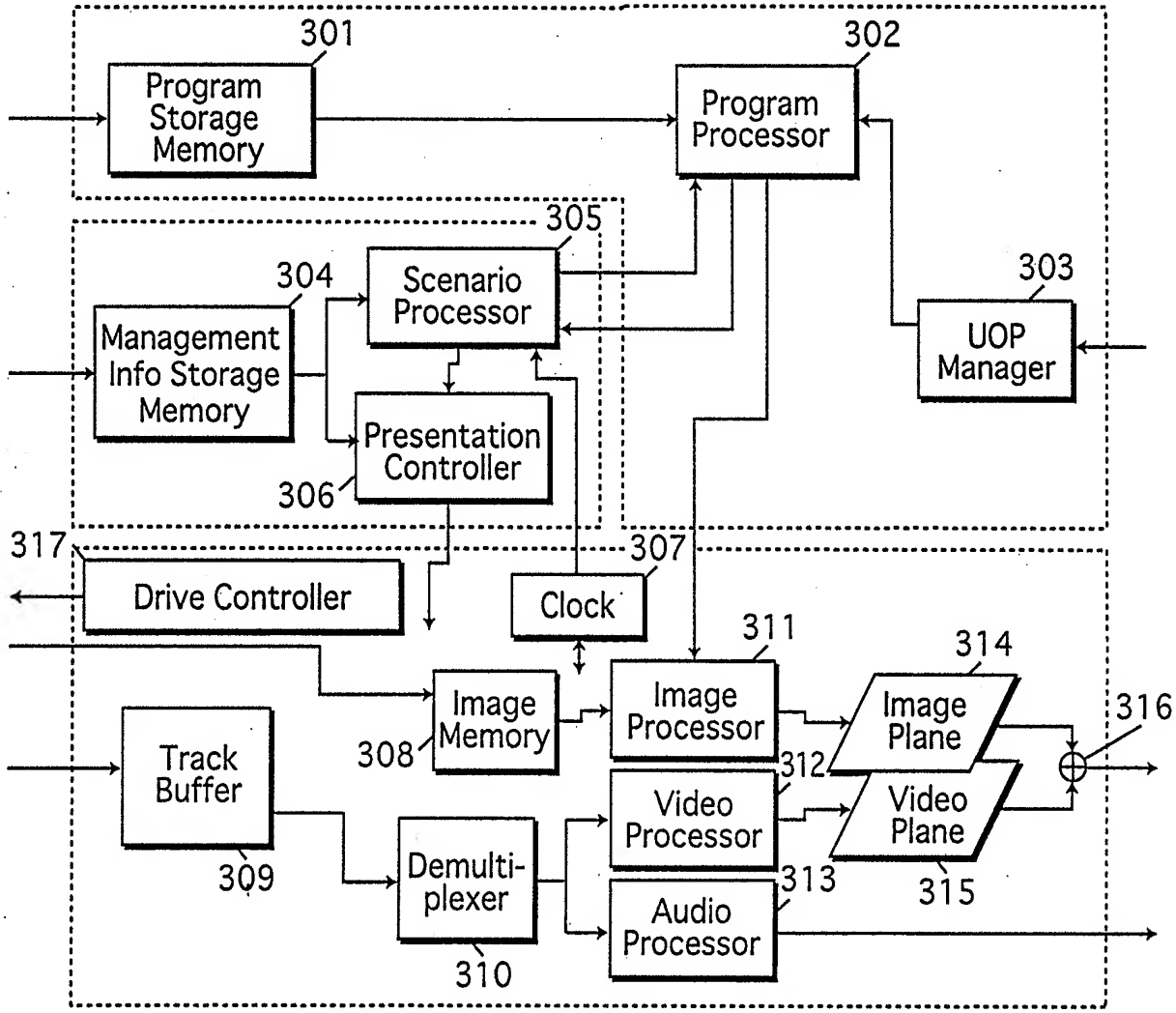

301
Program Storage Memory
302
Program Processor
305
304
Scenario Processor
303
Management Info Storage Memory
UOP Manager
Presentation Controller
306
317
307
Drive Controller
Clock
311
314
316
Image Memory
308
Image Processor
Image Plane
Track Buffer
312
Video Processor
Video Plane
309
Demulti-plexer
313
Audio Processor
315
310

FIG.8

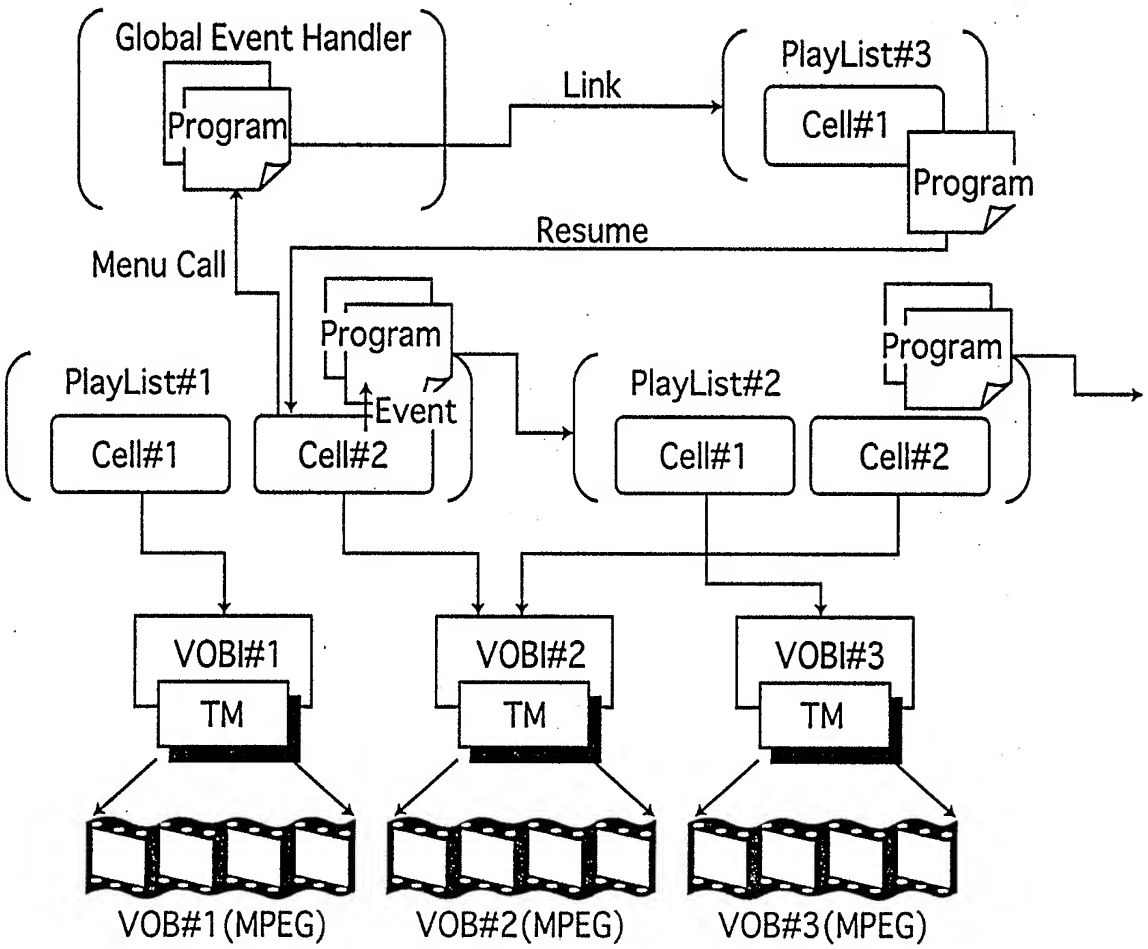

Global Event Handler
Program
Link
PlayList#3
Cell#1
Program
Resume
Menu Call
Program
Program
PlayList#1
Event
PlayList#2
Cell#1
Cell#2
Cell#1
Cell#2
VOBI#1
TM
VOBI#2
TM
VOBI#3
TM
VOB#1(MPEG)
VOB#2(MPEG)
VOB#3(MPEG)

FIG.9

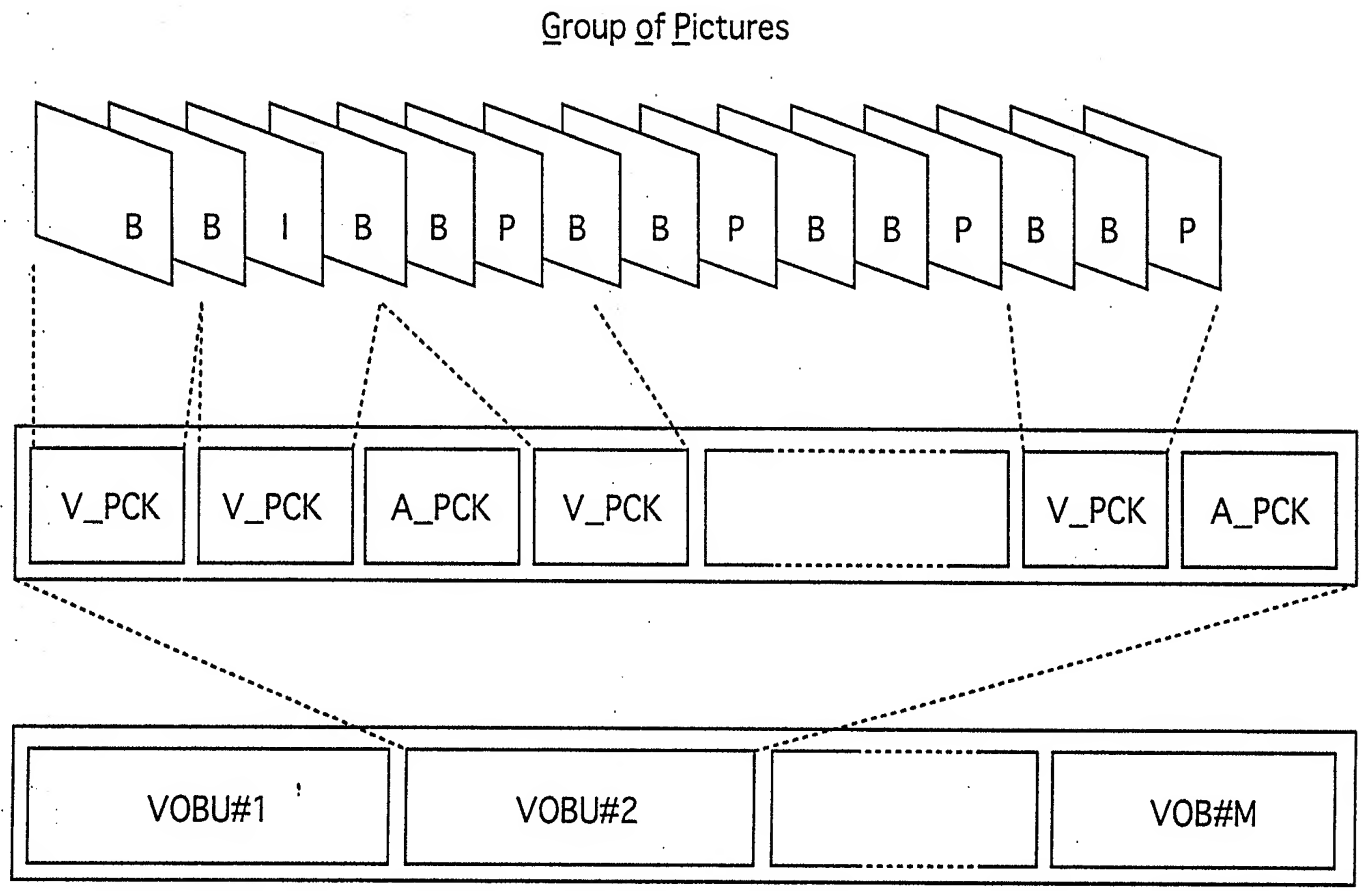

Group of Pictures
B
B
I
B
B
P
B
B
P
B
B
P
B
B
P
V_PCK
V_PCK
A_PCK
V_PCK
V_PCK
A_PCK
VOBU#1
VOBU#2
VOB#M

FIG.10

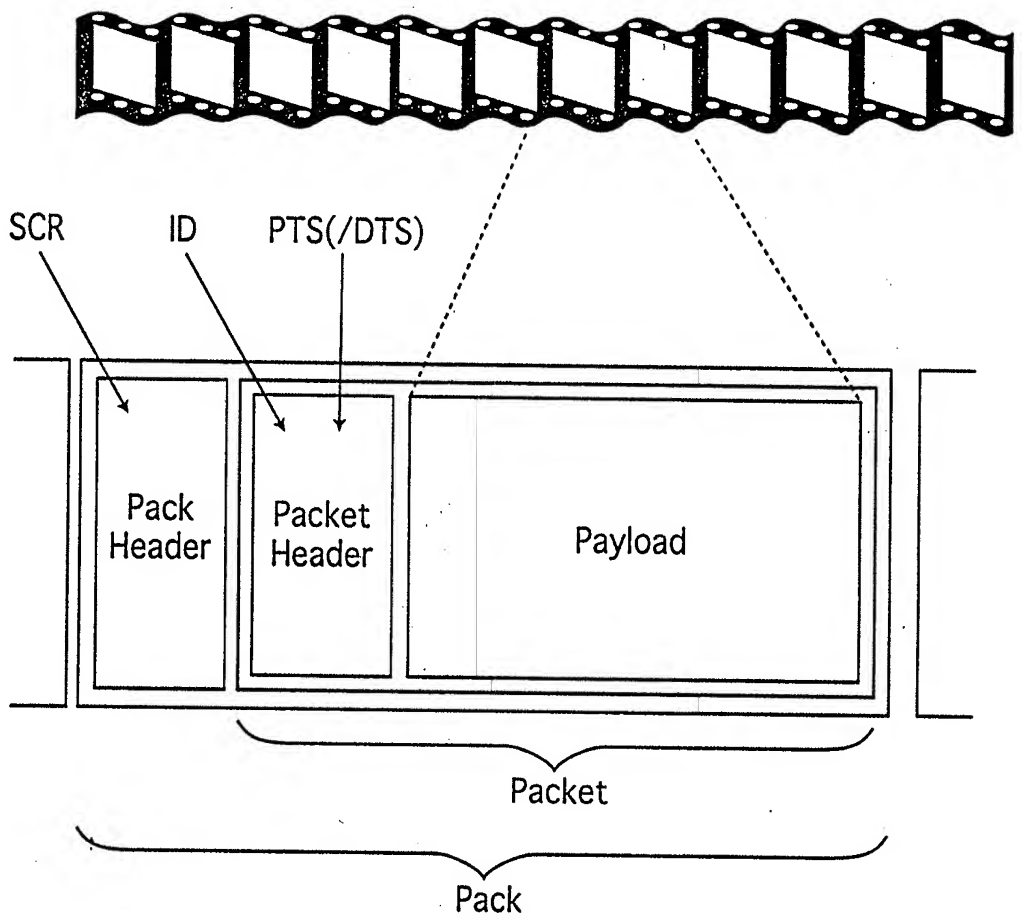

SCR
ID
PTS(/DTS)
Pack Header
Packet Header
Payload
Packet
Pack

FIG.11

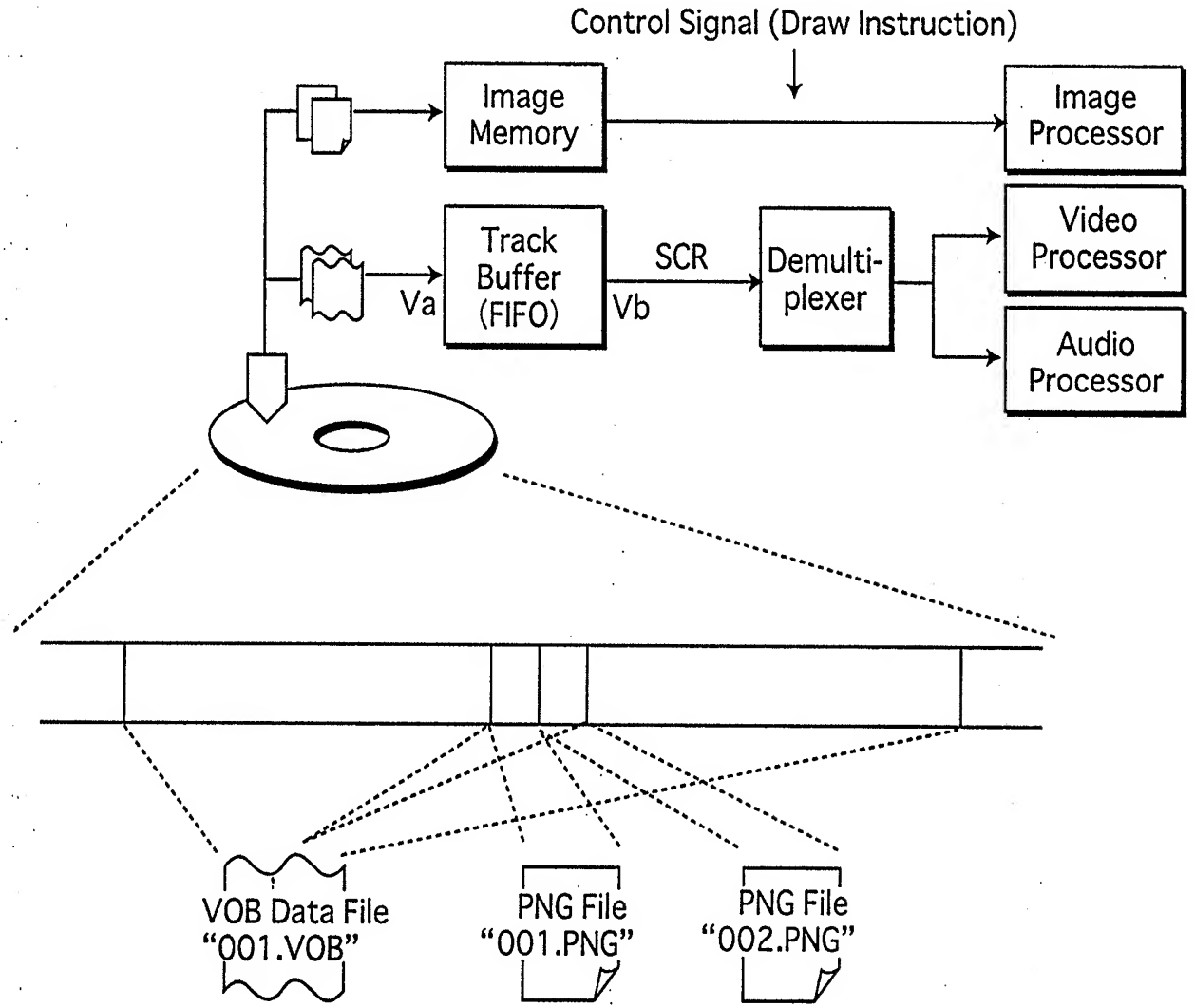

Control Signal (Draw Instruction)
Image Memory
Image Processor
Track Buffer (FIFO)
Va
Vb
SCR
Demulti-plexer
Video Processor
Audio Processor
VOB Data File "001.VOB"
PNG File "001.PNG"
PNG File "002.PNG"

FIG.12

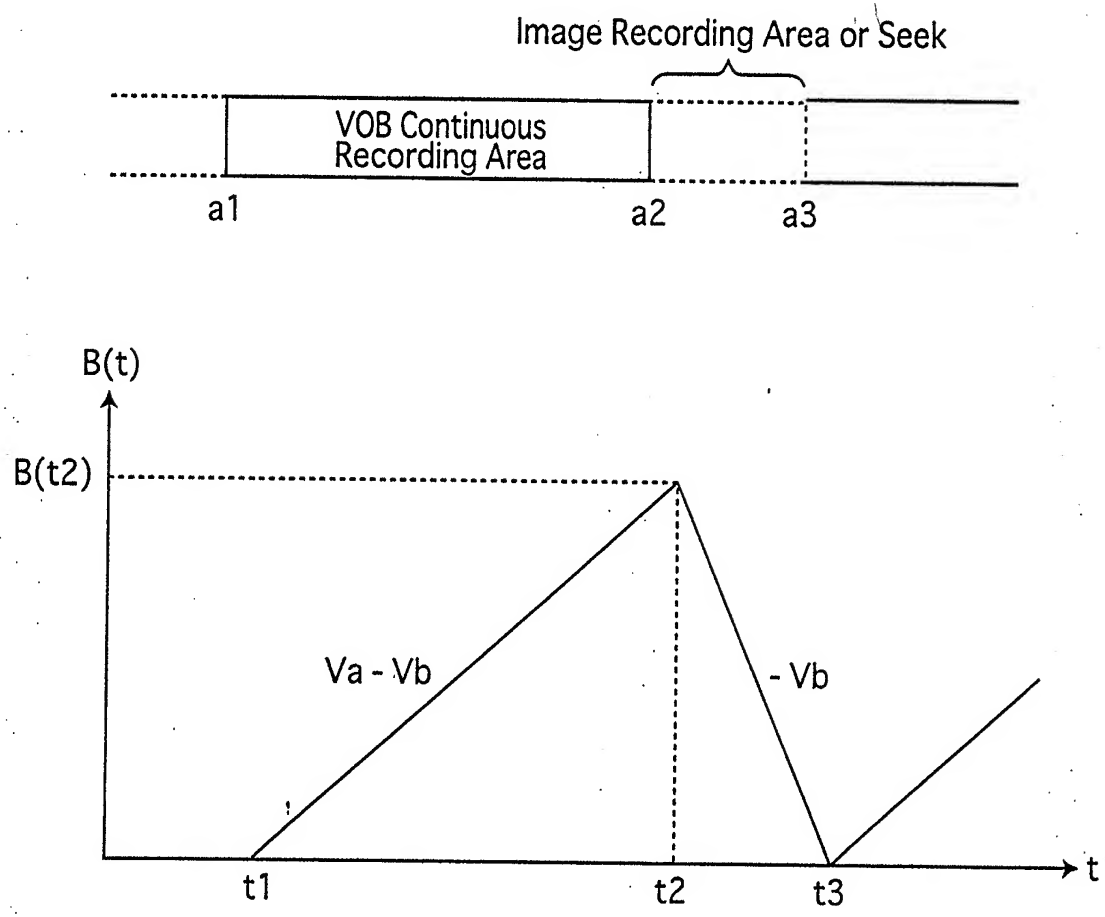


Image Recording Area or Seek
VOB Continuous Recording Area
a1
a2
a3
B(t)
B(t2)
Va - Vb
- Vb
t1
t2
t3
t

FIG.13

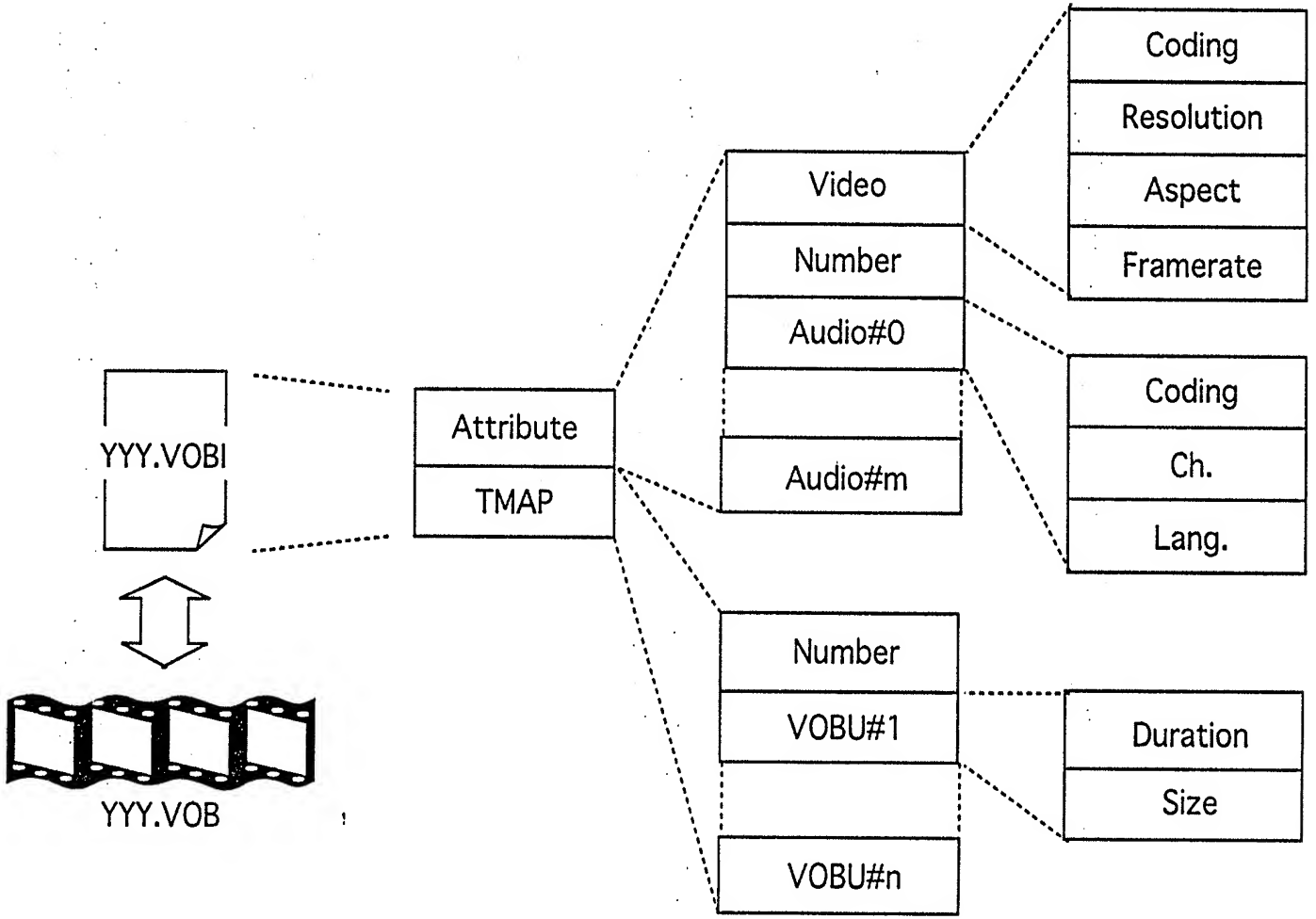

YYY.VOBI
YYY.VOB
Attribute
TMAP
Video
Number
Audio#0
Audio#m
Number
VOBU#1
VOBU#n
Coding
Resolution
Aspect
Framerate
Coding
Ch.
Lang.
Duration
Size

FIG.14

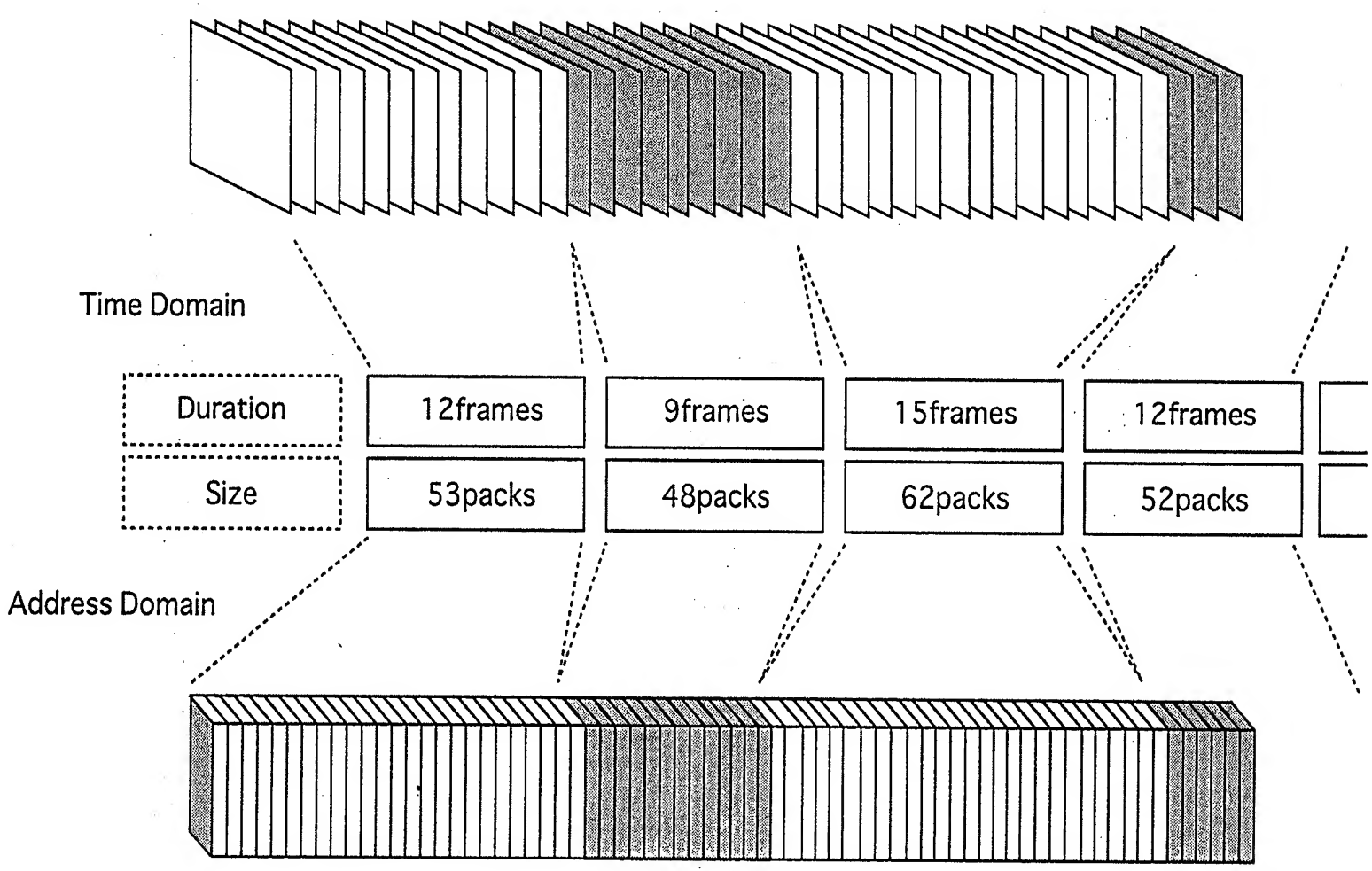

Time Domain
Duration
12frames
9frames
15frames
12frames
Size
53packs
48packs
62packs
52packs
Address Domain

FIG15

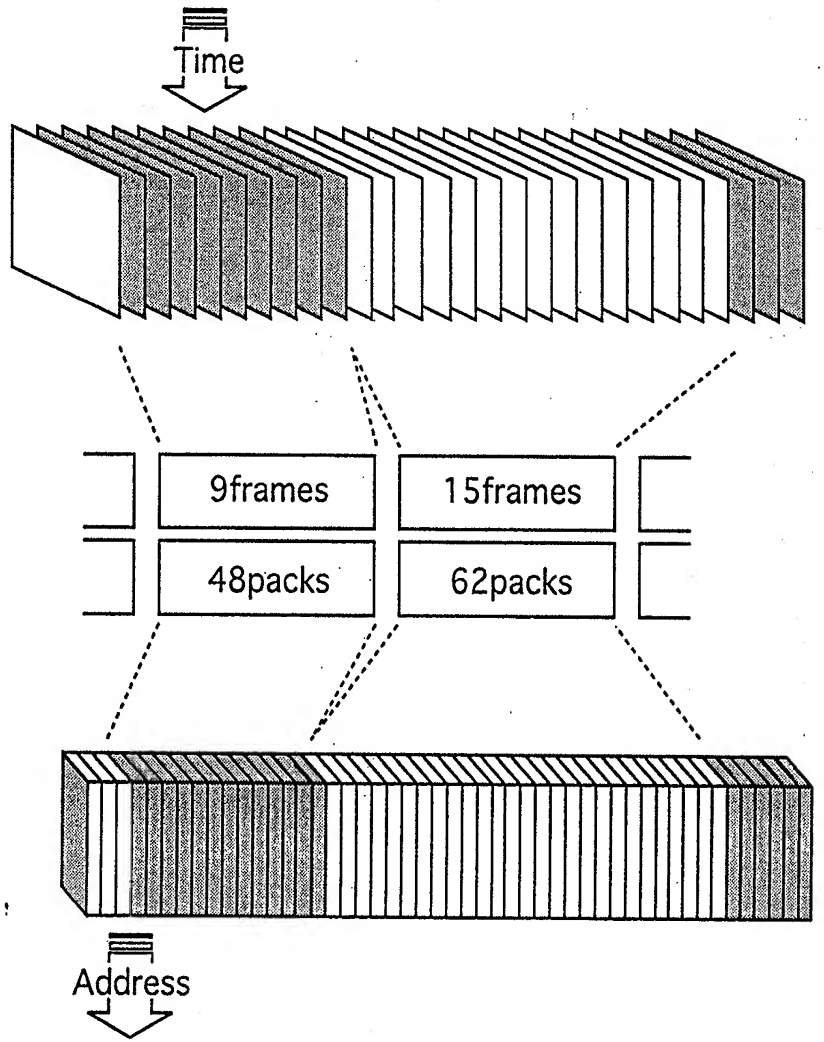


FIG.16

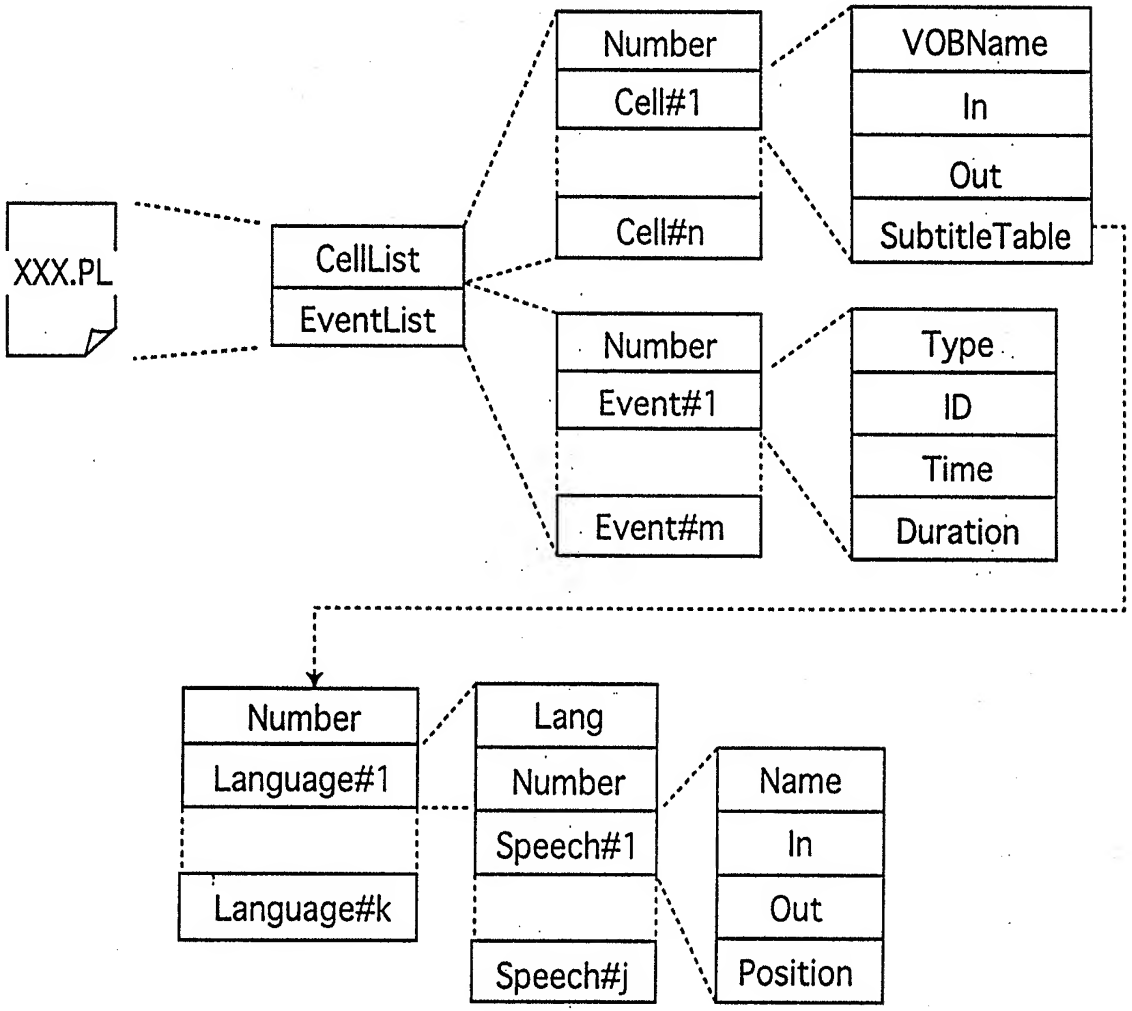

XXX.PL
CellList
EventList
Number
Cell#1
Cell#n
VOBName
In
Out
SubtitleTable
Number
Event#1
Event#m
Type
ID
Time
Duration
Number
Language#1
Language#k
Lang
Number
Speech#1
Speech#j
Name
In
Out
Position

FIG.17

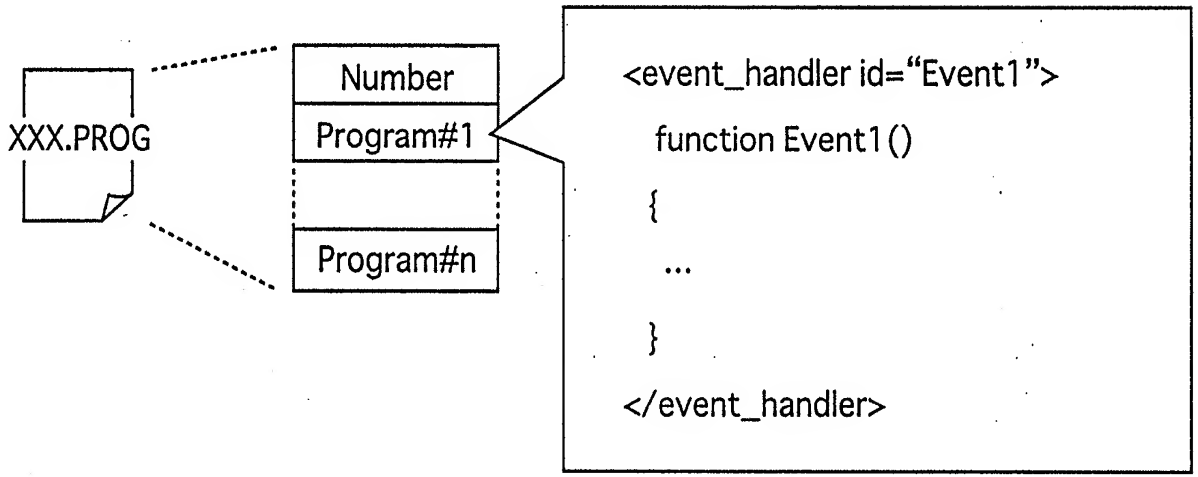

XXX.PROG
Number
Program#1
Program#n
<event_handler id="Event1">
function Event1 ()
{
...
}
</event_handler>

FIG.18

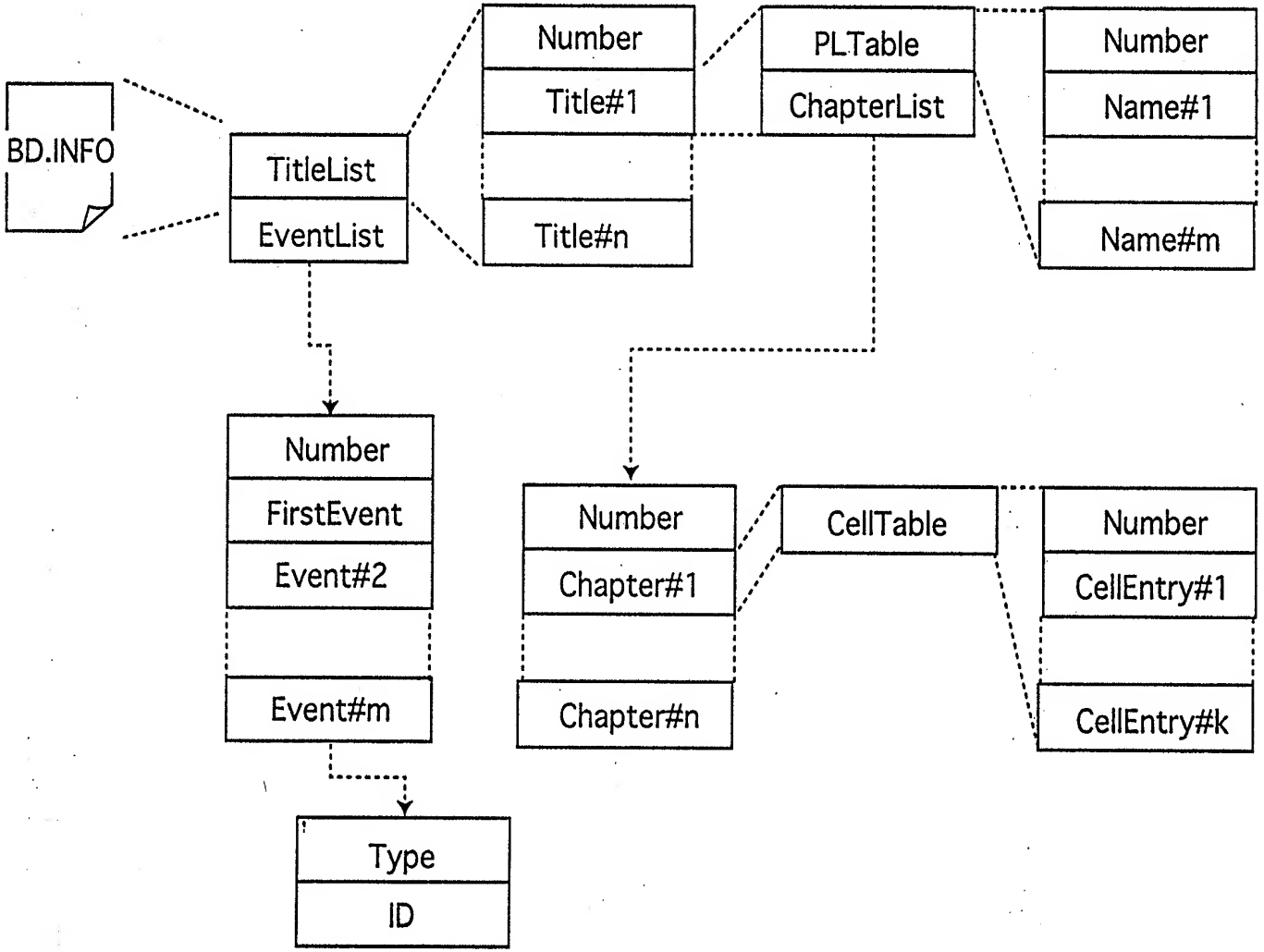

BD.INFO
TitleList
EventList
Number
Title#1
Title#n
PLTable
ChapterList
Number
Name#1
Name#m
Number
FirstEvent
Event#2
Event#m
Number
Chapter#1
Chapter#n
CellTable
Number
CellEntry#1
CellEntry#k
Type
ID

FIG.19

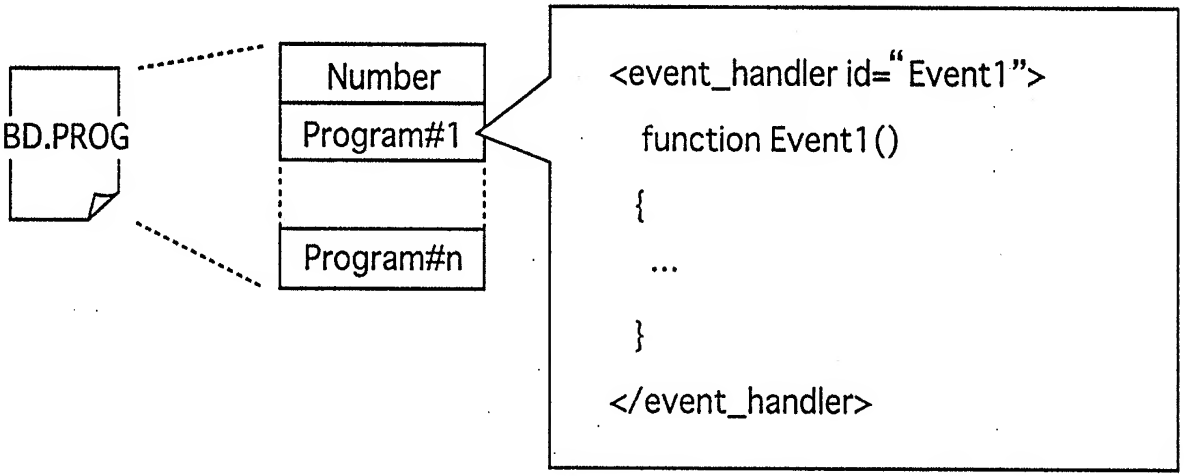

BD.PROG
Number
Program#1
Program#n
<event_handler id="Event1">
function Event1()
{
...
}
</event_handler>

FIG.20

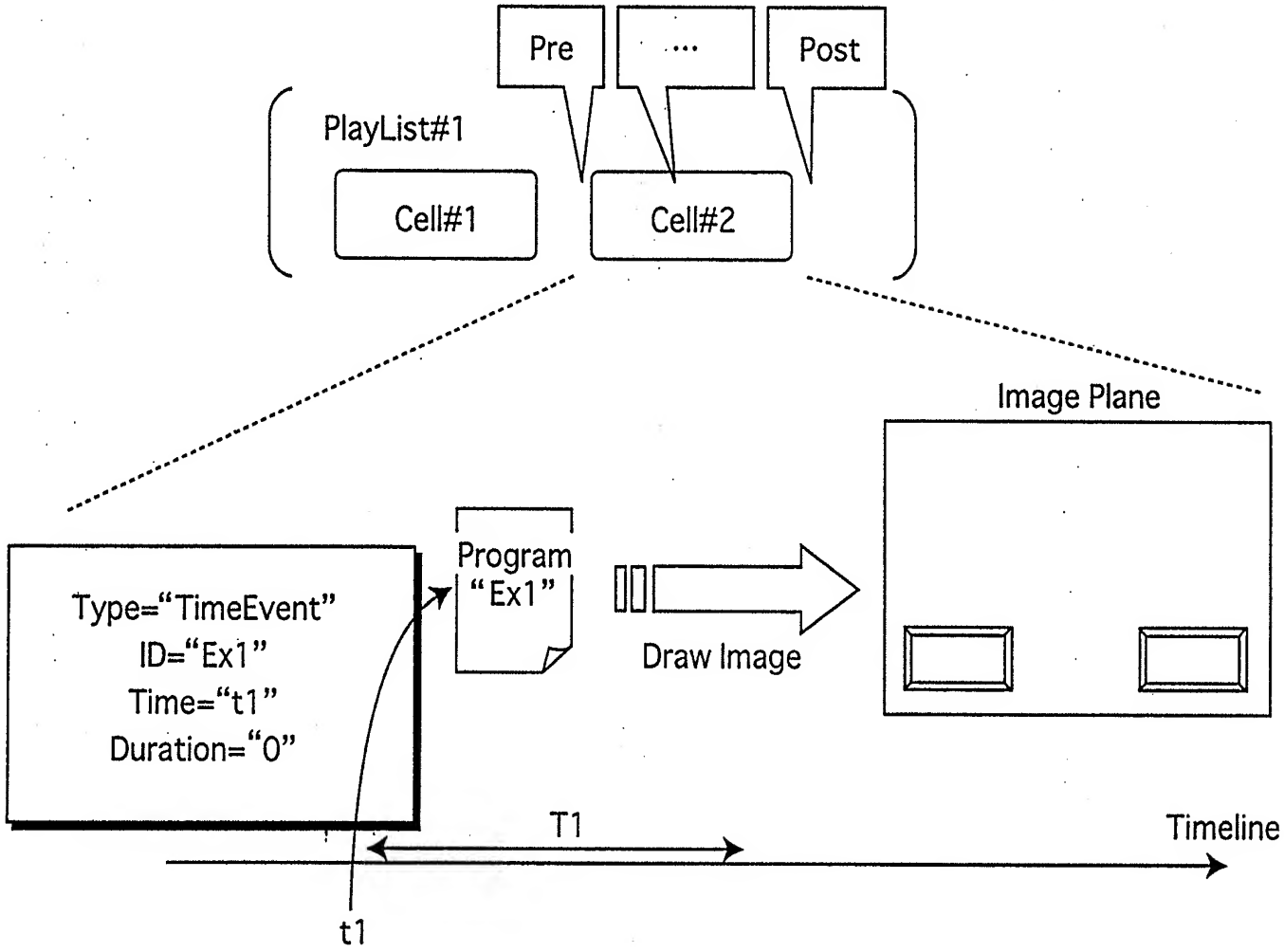

Pre
...
Post
PlayList#1
Cell#1
Cell#2
Image Plane
Type="TimeEvent"
ID="Ex1"
Time="t1"
Duration="0"
Program
"Ex1"
Draw Image
T1
Timeline
t1

FIG.21

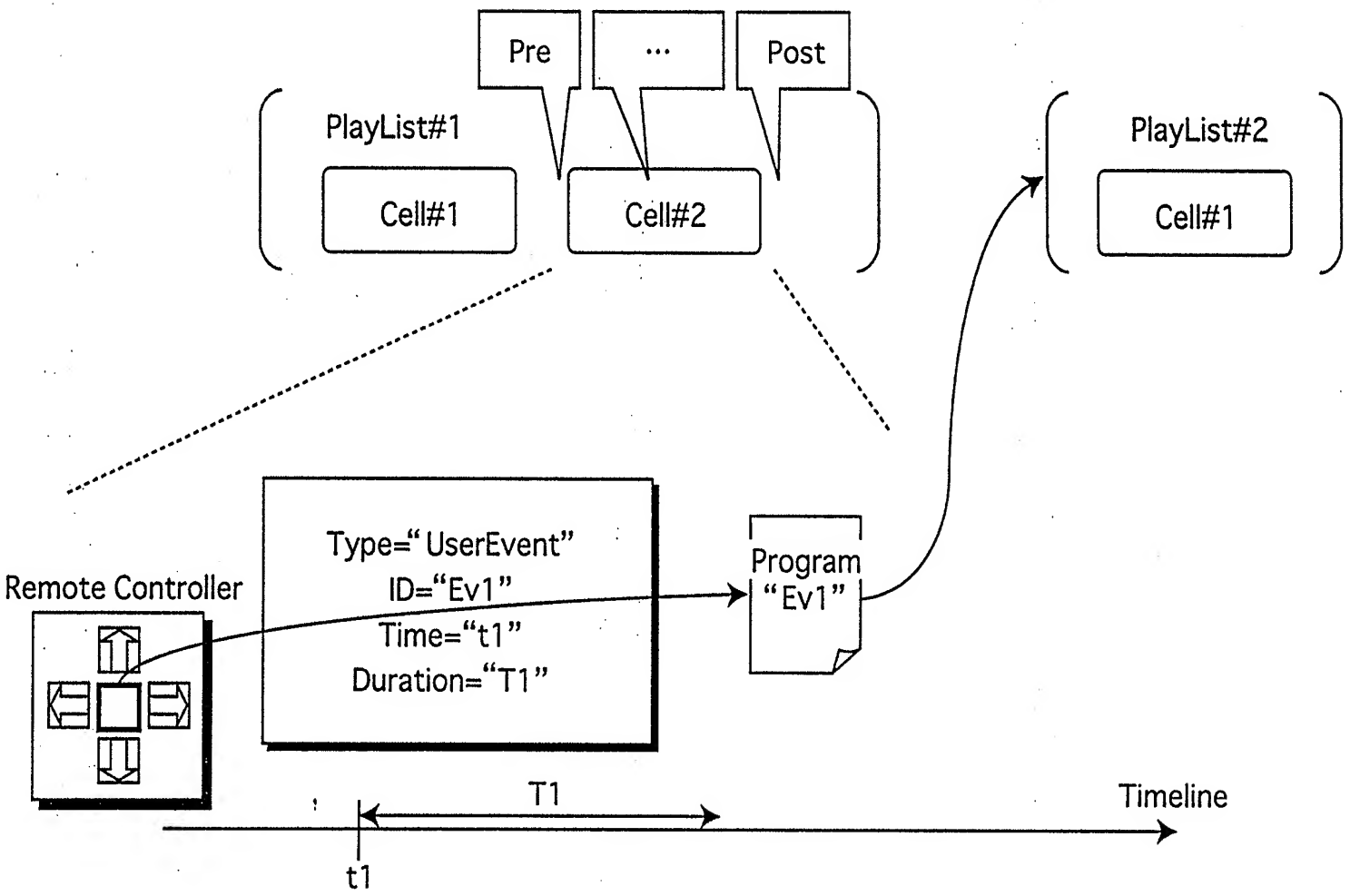

Pre
...
Post
PlayList#1
Cell#1
Cell#2
PlayList#2
Cell#1
Type="UserEvent"
ID="Ev1"
Time="t1"
Duration="T1"
Remote Controller
Program
"Ev1"
T1
Timeline
t1

FIG.22

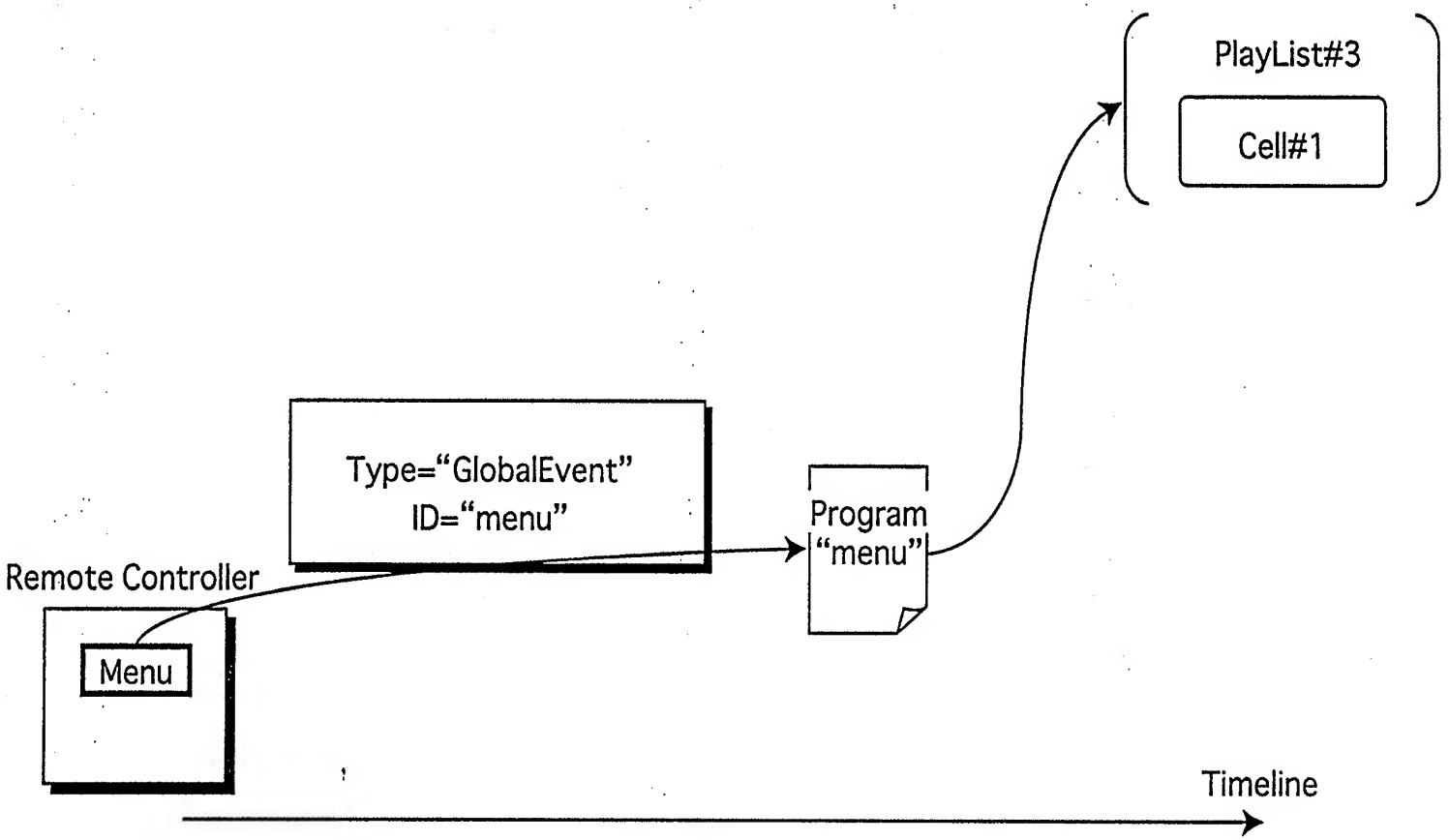

PlayList#3
Cell#1
Type="GlobalEvent"
ID="menu"
Program
"menu"
Remote Controller
Menu
Timeline

FIG.23

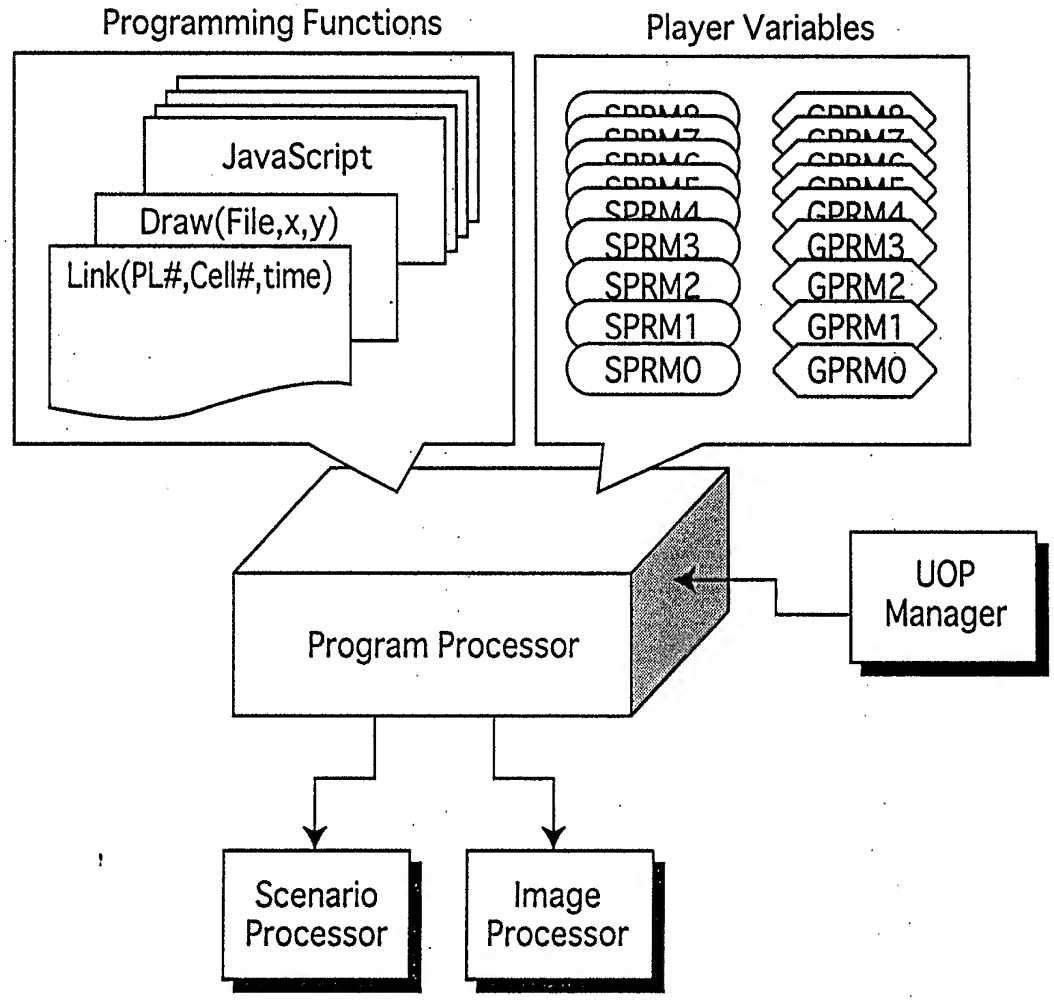

Programming Functions
Player Variables
JavaScript
Draw(File,x,y)
Link(PL#,Cell#,time)
SPRM3
SPRM2
SPRM1
SPRM0
GPRM3
GPRM2
GPRM1
GPRM0
Program Processor
UOP Manager
Scenario Processor
Image Processor

FIG.24

Player Variables (System Parameters)

0	Language Code
1	Audio stream number
2	Subtitle stream number
3	Angle number
4	Title number
5	Chapter number
6	Program number
7	Cell number
8	Key name
9	Navigation timer
10	Current playback time

11	Player audio mixing mode for Karaoke
12	Country code for parental management
13	Parental level
14	Player configuration for Video
15	Player configuration for Audio
16	Language code for AST
17	Language code ext. for AST
18	Language code for STST
19	Language coded ext. for STST
20	Player region code
21	reserved

22	reserved
23	Player status
24	reserved
25	reserved
26	reserved
27	reserved
28	reserved
29	reserved
30	reserved
31	reserved
32	reserved

FIG.25

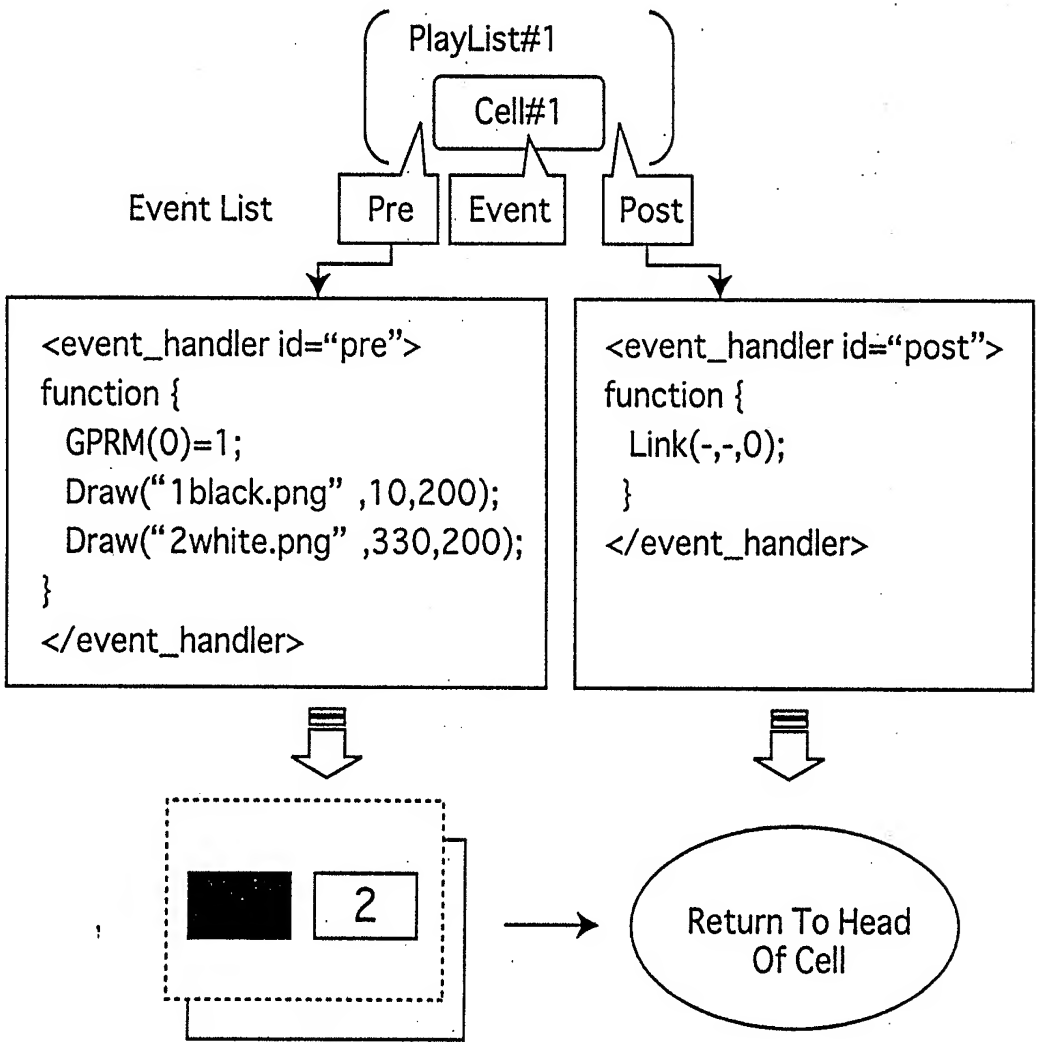

PlayList#1
Cell#1
Event List
Pre
Event
Post
<event_handler id="pre">
function {
GPRM(0)=1;
Draw("1black.png" ,10,200);
Draw("2white.png" ,330,200);
}
</event_handler>
<event_handler id="post">
function {
Link(-,-,0);
}
</event_handler>
2
Return To Head Of Cell

FIG.26

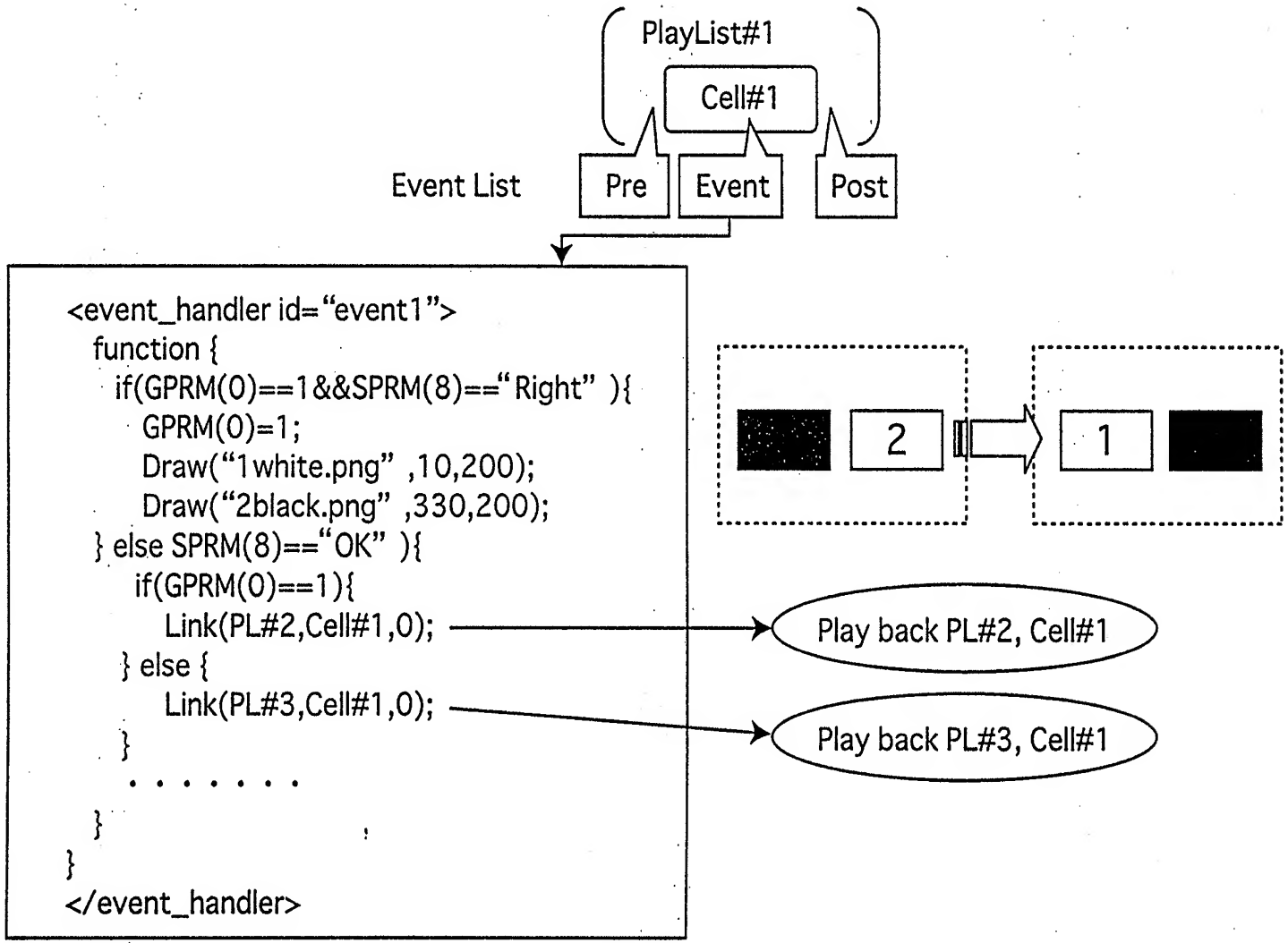

PlayList#1
Cell#1
Event List
Pre
Event
Post
<event_handler id="event1">
function {
if(GPRM(0)==1&&SPRM(8)=="Right"){
GPRM(0)=1;
Draw("1white.png" ,10,200);
Draw("2black.png" ,330,200);
} else SPRM(8)=="OK"){
if(GPRM(0)==1){
Link(PL#2,Cell#1,0);
} else {
Link(PL#3,Cell#1,0);
}
. . . . . .
}
}
</event_handler>
2
1
Play back PL#2, Cell#1
Play back PL#3, Cell#1

FIG.27

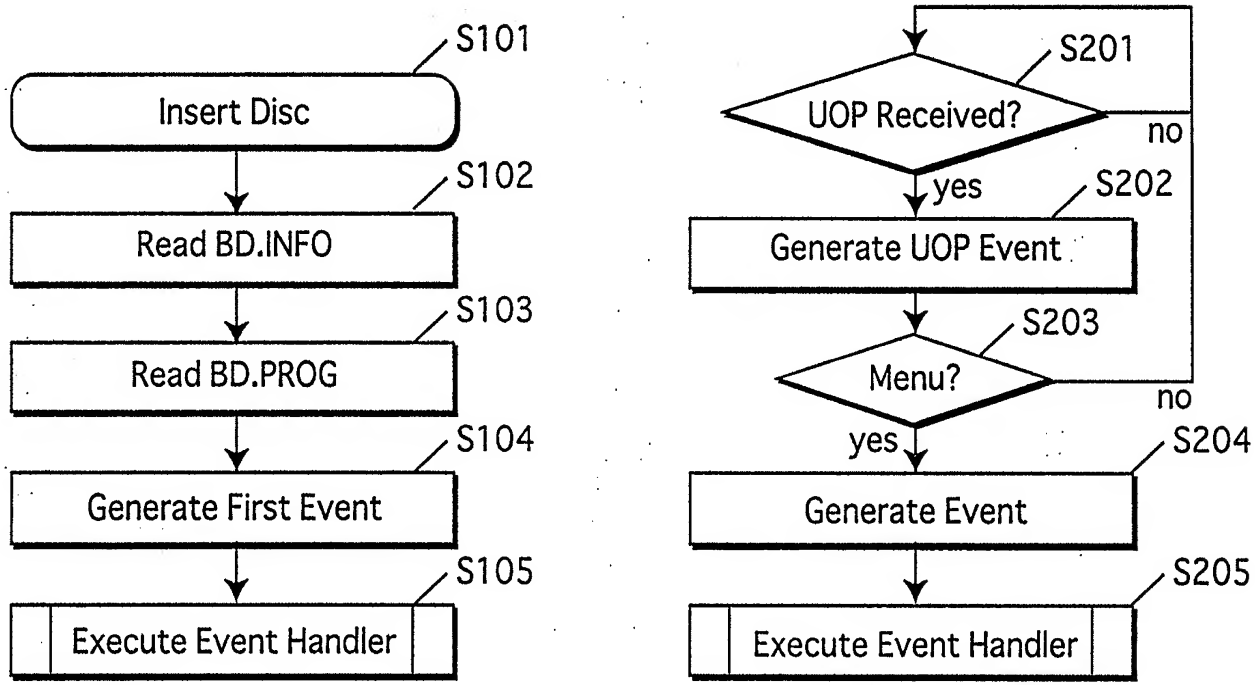

S101
Insert Disc
S102
Read BD.INFO
S103
Read BD.PROG
S104
Generate First Event
S105
Execute Event Handler
S201
UOP Received?
no
yes
S202
Generate UOP Event
S203
Menu?
no
yes
S204
Generate Event
S205
Execute Event Handler

FIG.28

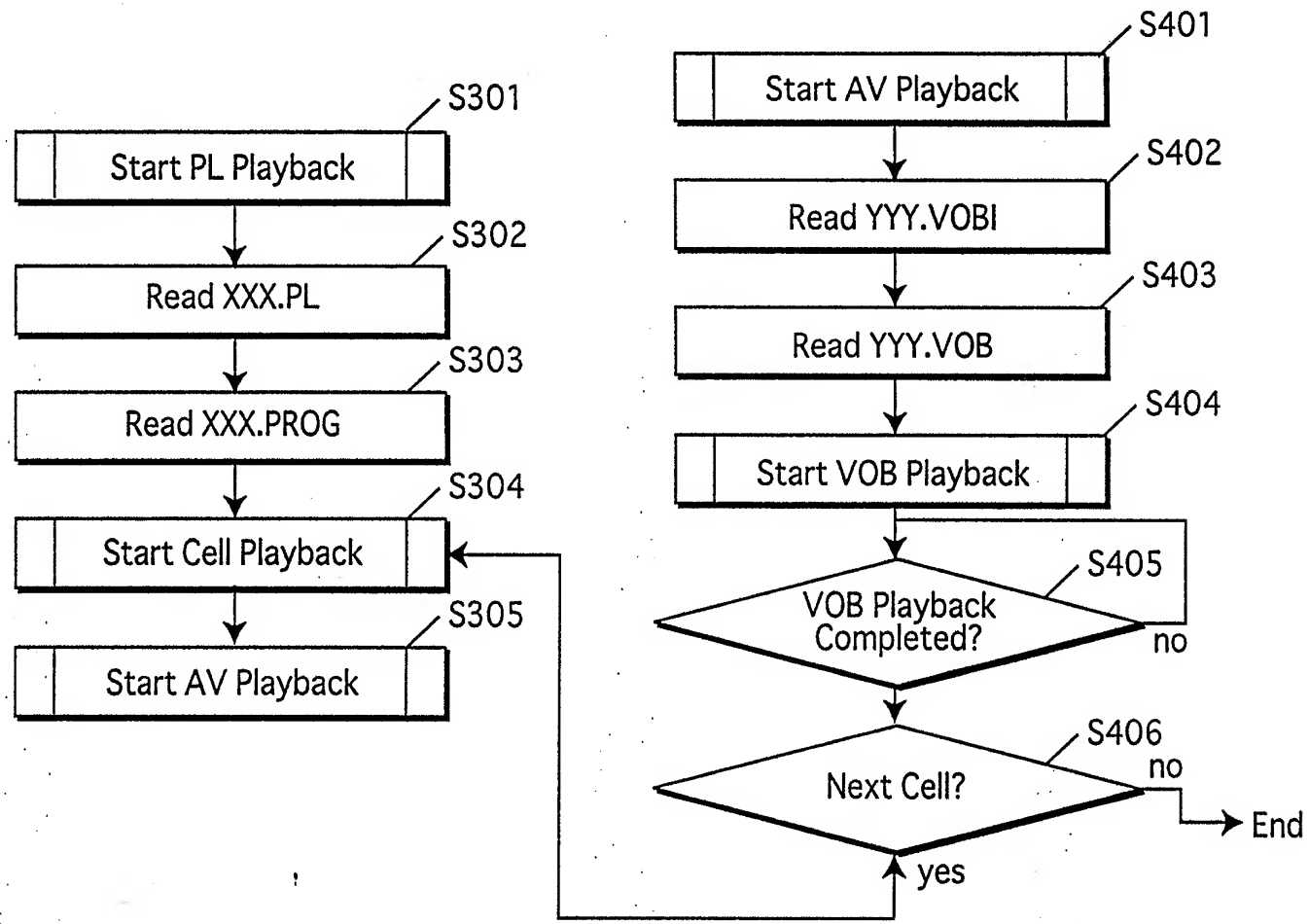
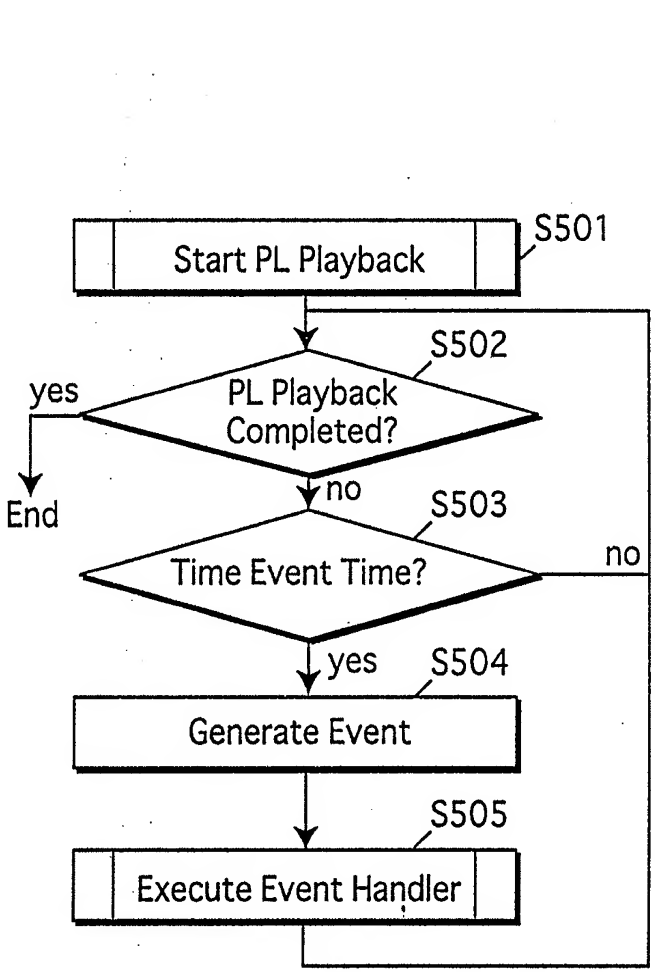

S301
Start PL Playback
S302
Read XXX.PL
S303
Read XXX.PROG
S304
Start Cell Playback
S305
Start AV Playback
S401
Start AV Playback
S402
Read YYY.VOBI
S403
Read YYY.VOB
S404
Start VOB Playback
S405
VOB Playback Completed?
no
S406
Next Cell?
no
End
yes

FIG.29


Start PL Playback
S501
S502
PL Playback Completed?
yes
End
no
S503
Time Event Time?
no
yes
S504
Generate Event
S505
Execute Event Handler

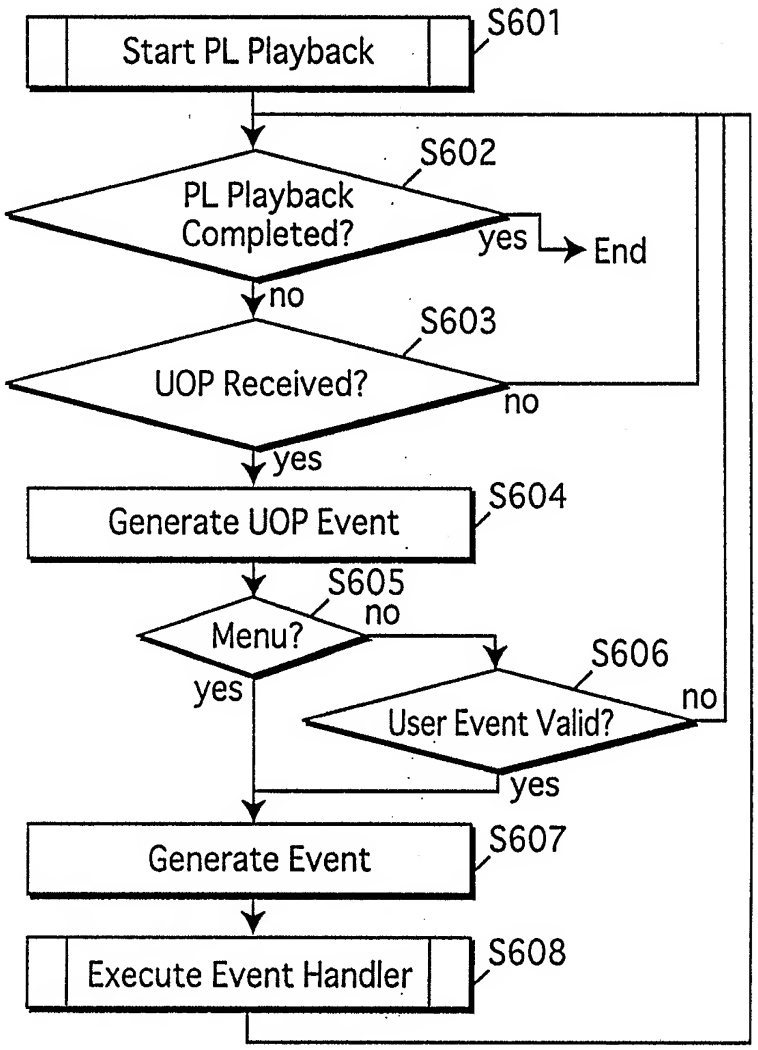

Start PL Playback
S601
S602
PL Playback Completed?
yes
End
no
S603
UOP Received?
no
yes
S604
Generate UOP Event
S605
Menu?
no
yes
S606
User Event Valid?
no
yes
S607
Generate Event
S608
Execute Event Handler

FIG.30

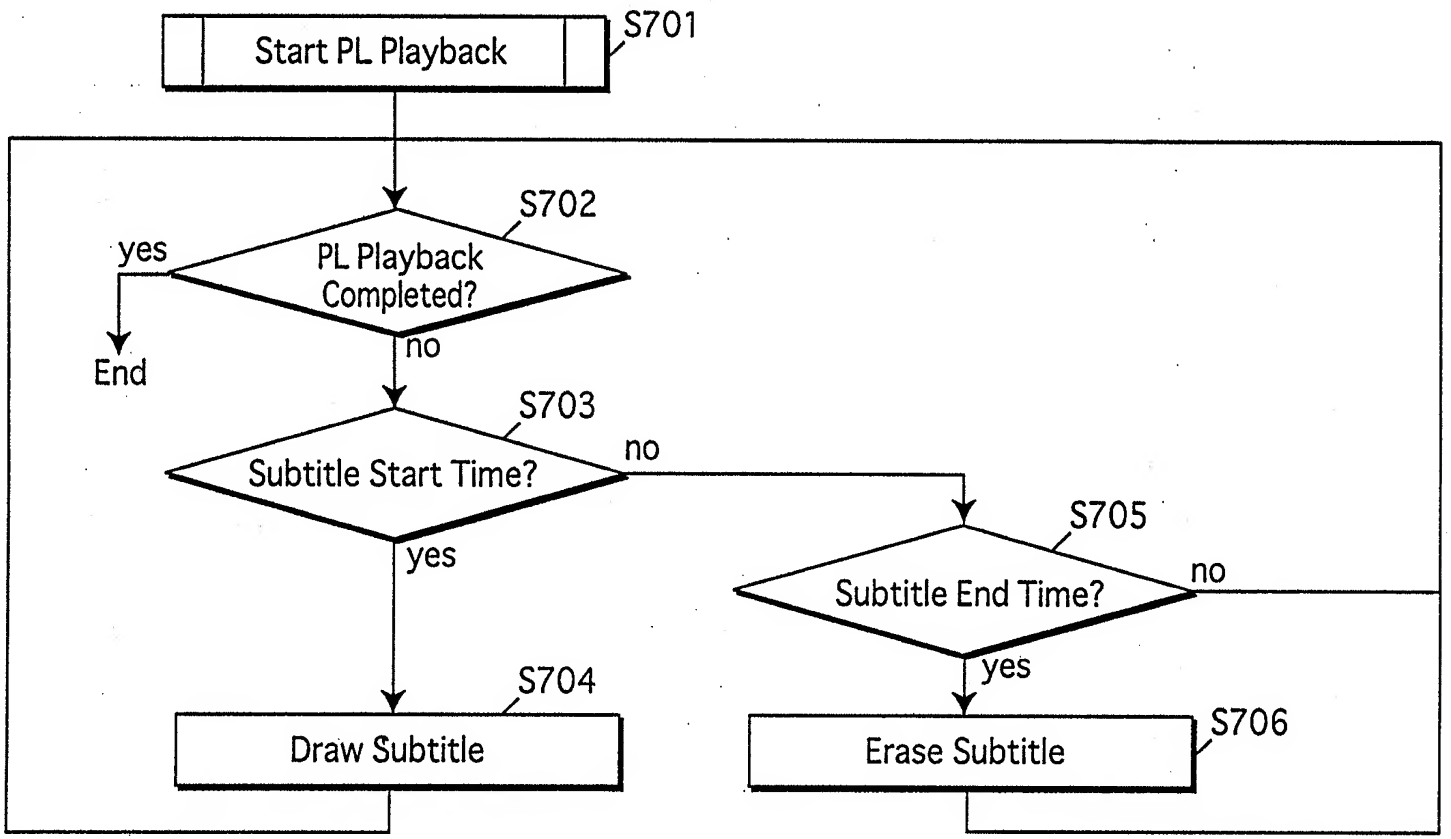

Start PL Playback
S701
S702
PL Playback Completed?
yes
End
no
S703
Subtitle Start Time?
no
yes
S704
Draw Subtitle
S705
Subtitle End Time?
no
yes
S706
Erase Subtitle

FIG.31

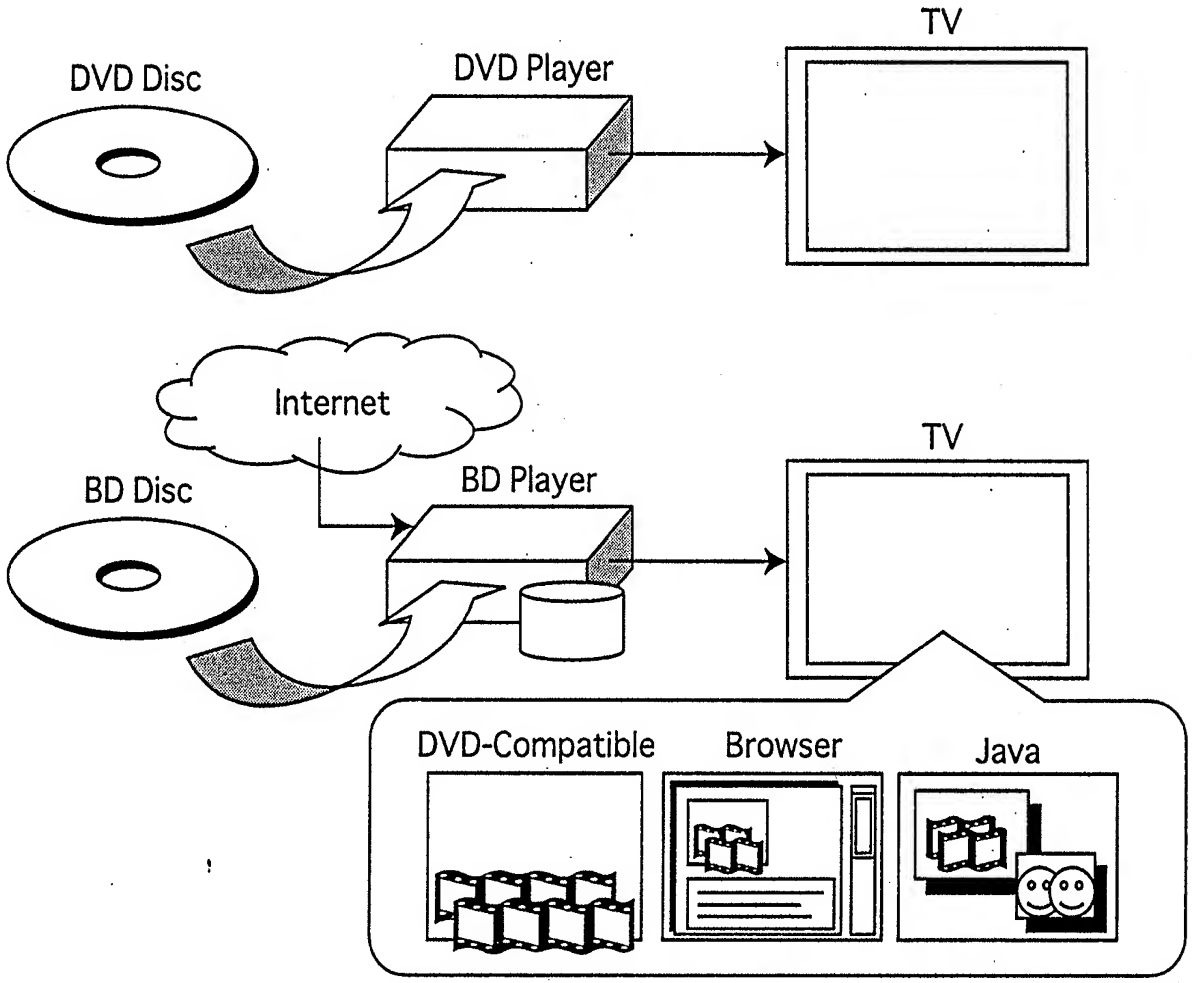

TV
DVD Disc
DVD Player
Internet
TV
BD Disc
BD Player
DVD-Compatible
Browser
Java

FIG.32

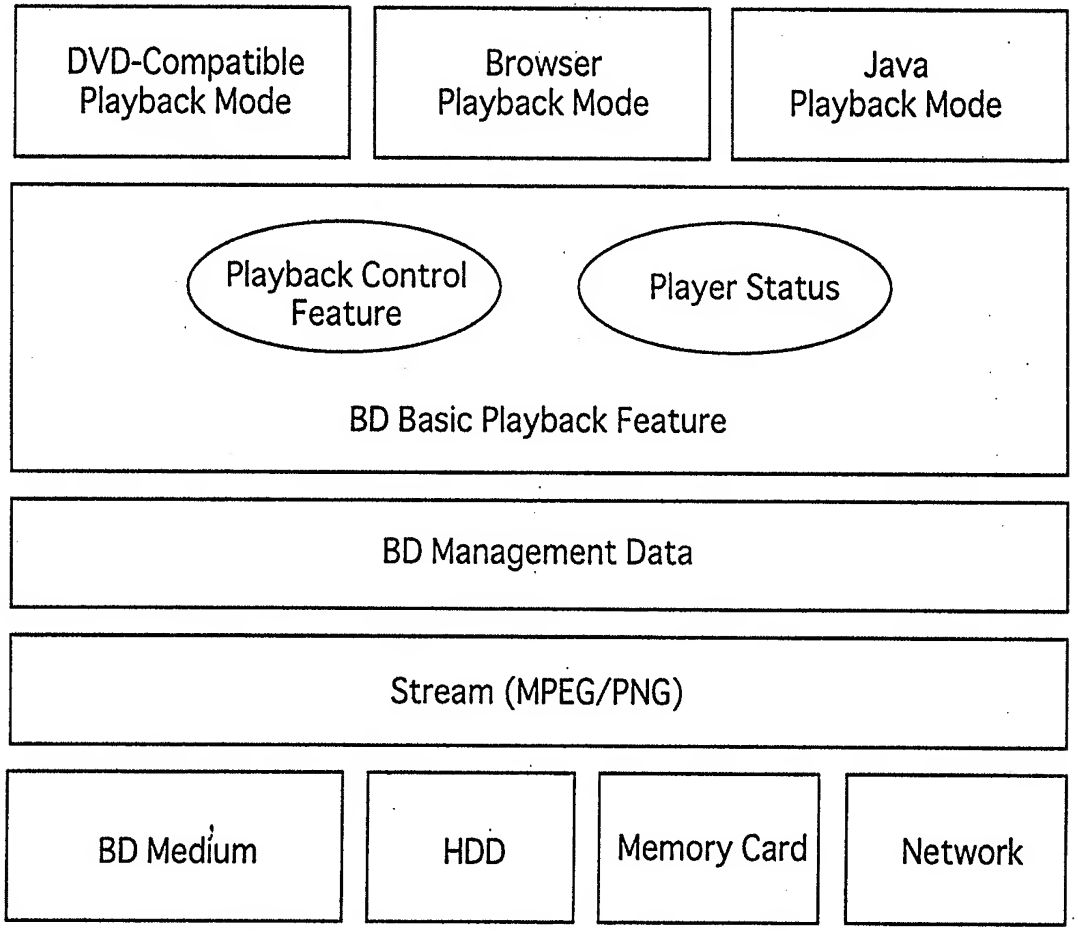

DVD-Compatible Playback Mode
Browser Playback Mode
Java Playback Mode
Playback Control Feature
Player Status
BD Basic Playback Feature
BD Management Data
Stream (MPEG/PNG)
BD Medium
HDD
Memory Card
Network

FIG.33

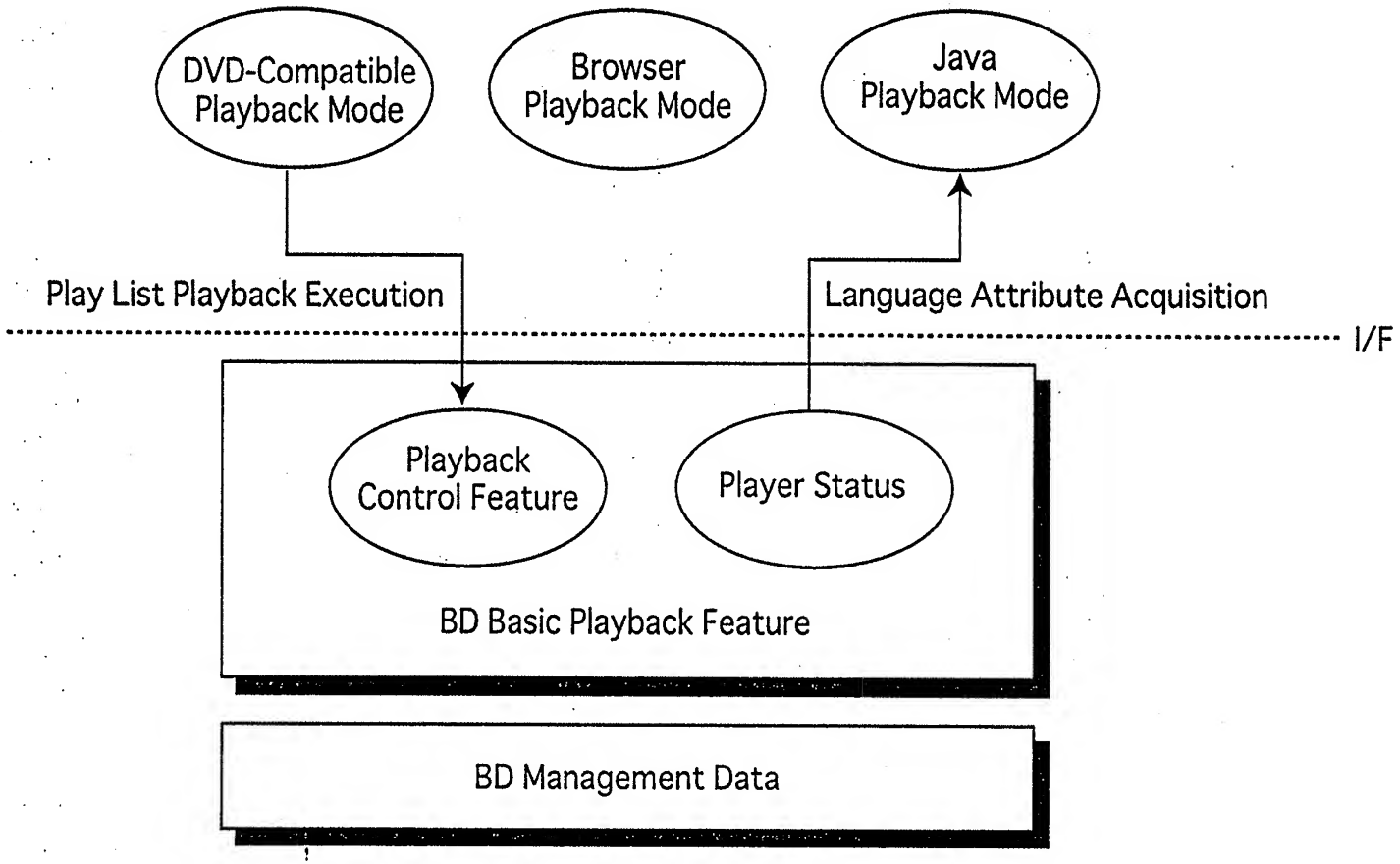

DVD-Compatible Playback Mode
Browser Playback Mode
Java Playback Mode
Play List Playback Execution
Language Attribute Acquisition
I/F
Playback Control Feature
Player Status
BD Basic Playback Feature
BD Management Data

FIG.34

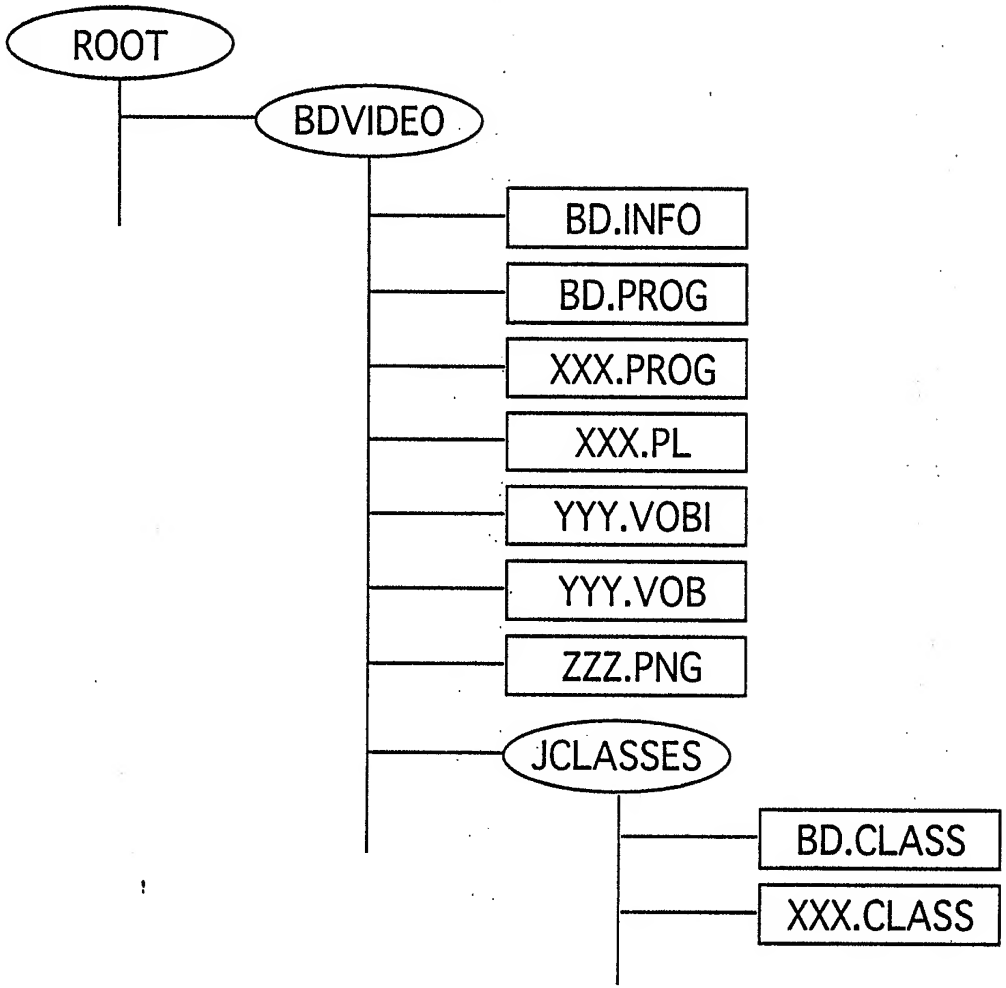

ROOT
BDVIDEO
BD.INFO
BD.PROG
XXX.PROG
XXX.PL
YYY.VOBI
YYY.VOB
ZZZ.PNG
JCLASSES
BD.CLASS
XXX.CLASS

FIG.35

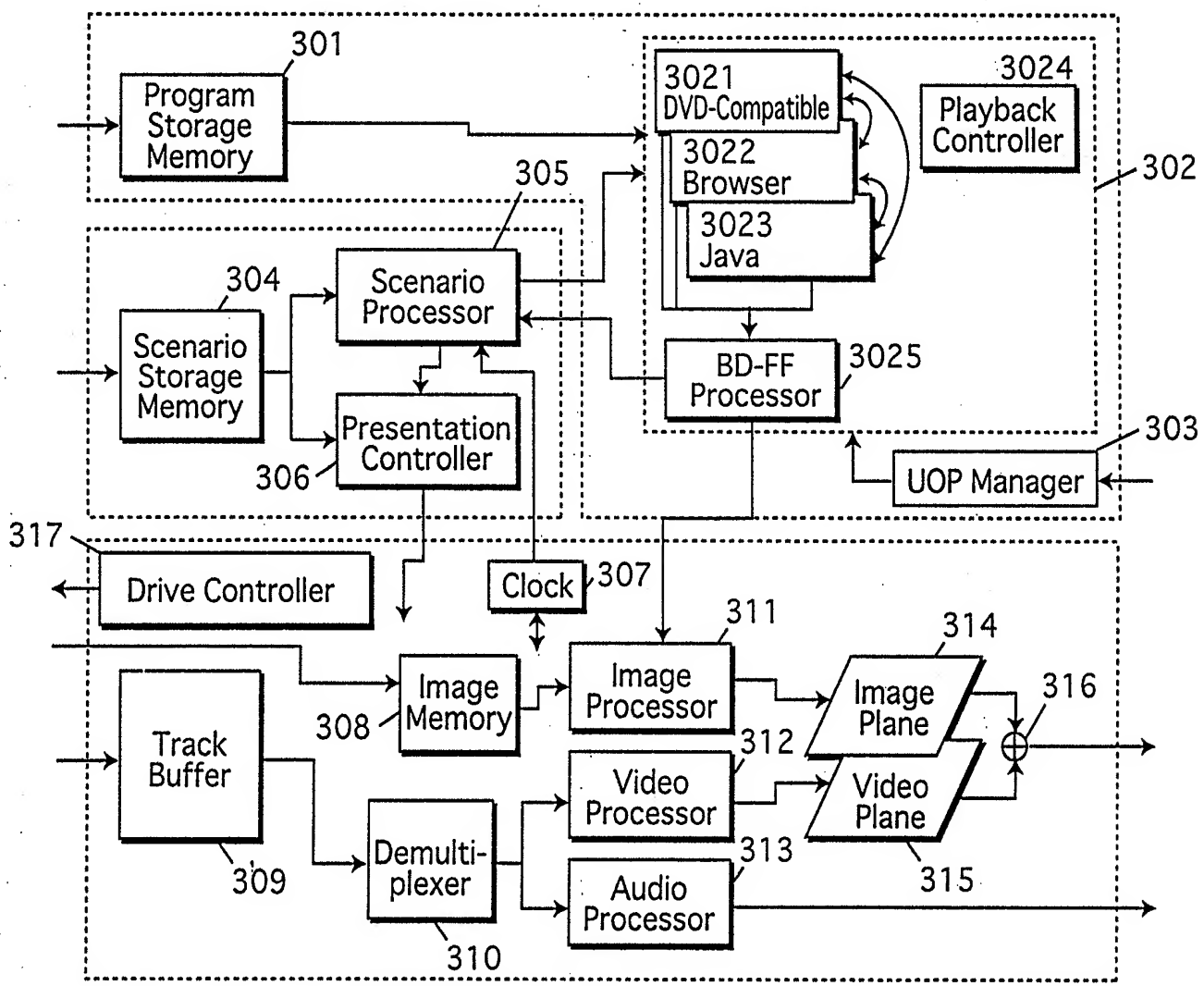

301
Program Storage Memory
3021 DVD-Compatible
3022 Browser
3023 Java
3024 Playback Controller
302
305
304
Scenario Processor
Scenario Storage Memory
Presentation Controller
306
BD-FF Processor
3025
303
UOP Manager
317
Drive Controller
Clock
307
311
314
316
Image Memory
308
Image Processor
Image Plane
312
Track Buffer
Video Processor
Video Plane
313
Demulti-plexer
Audio Processor
309
310
315

FIG.36

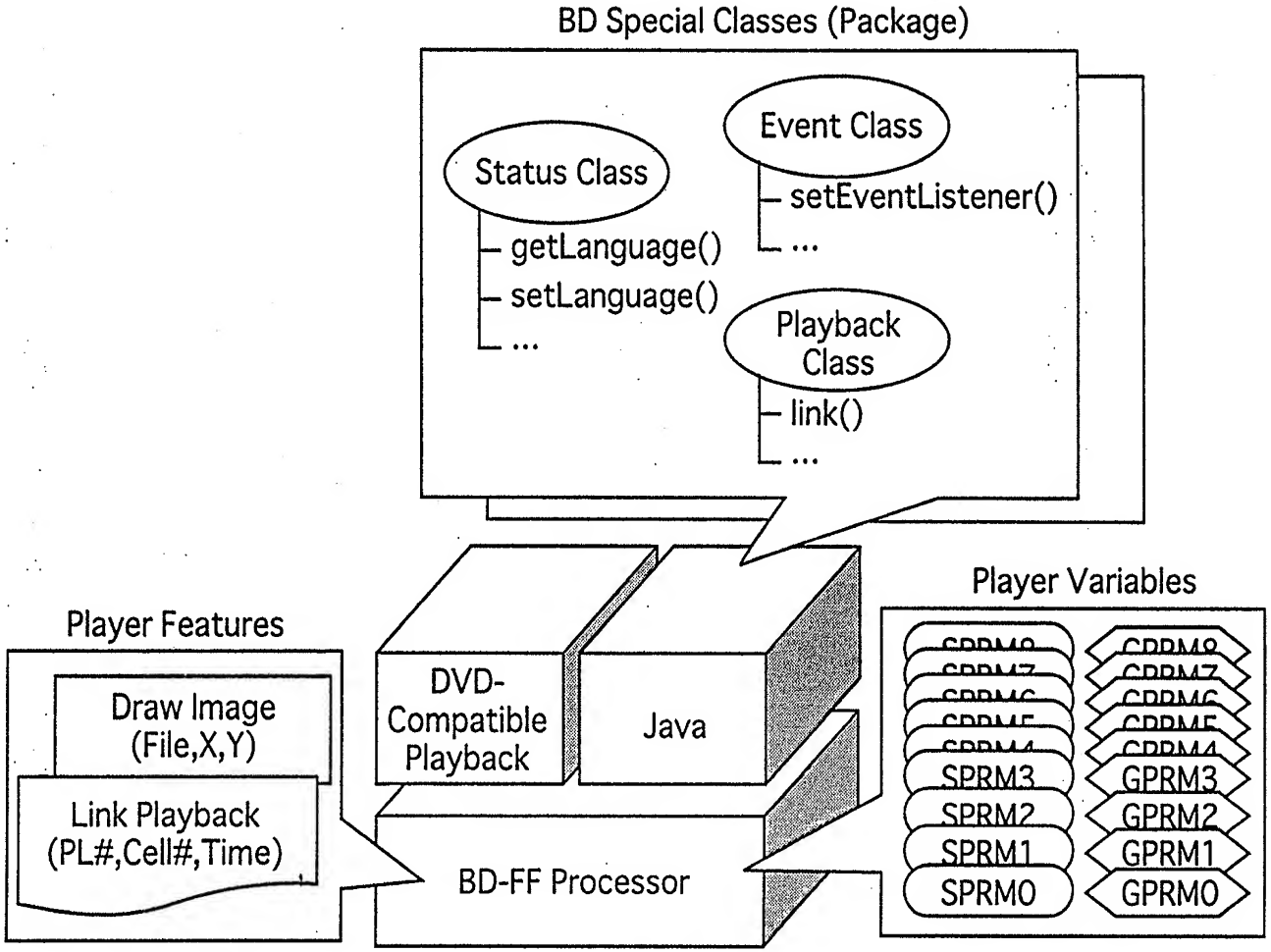

BD Special Classes (Package)
Status Class
getLanguage()
setLanguage()
...
Event Class
setEventListener()
...
Playback Class
link()
...
Player Features
Draw Image (File,X,Y)
Link Playback (PL#,Cell#,Time)
DVD-Compatible Playback
Java
BD-FF Processor
Player Variables
SPRM3
SPRM2
SPRM1
SPRM0
GPRM3
GPRM2
GPRM1
GPRM0

FIG.37

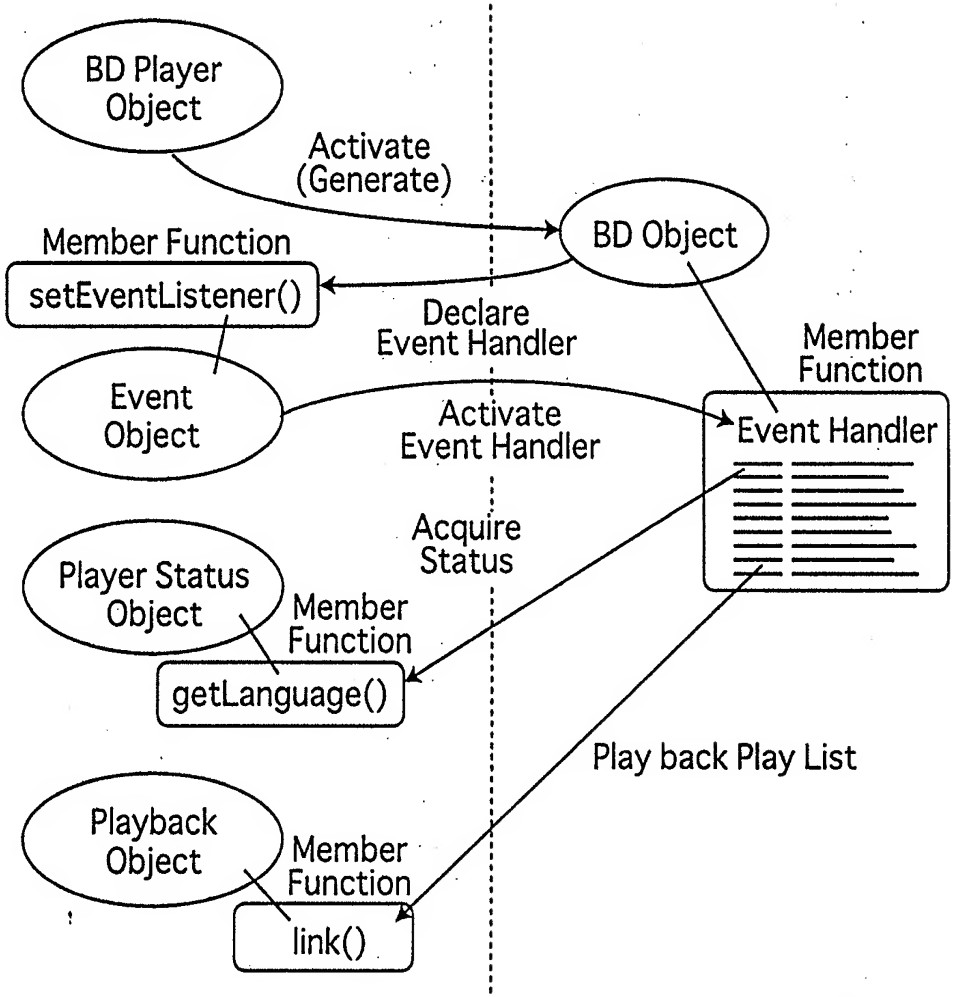

BD Player Object
Activate (Generate)
BD Object
Member Function
setEventListener()
Declare Event Handler
Member Function
Event Object
Activate Event Handler
Event Handler
Acquire Status
Player Status Object
Member Function
getLanguage()
Play back Play List
Playback Object
Member Function
link()

FIG.38

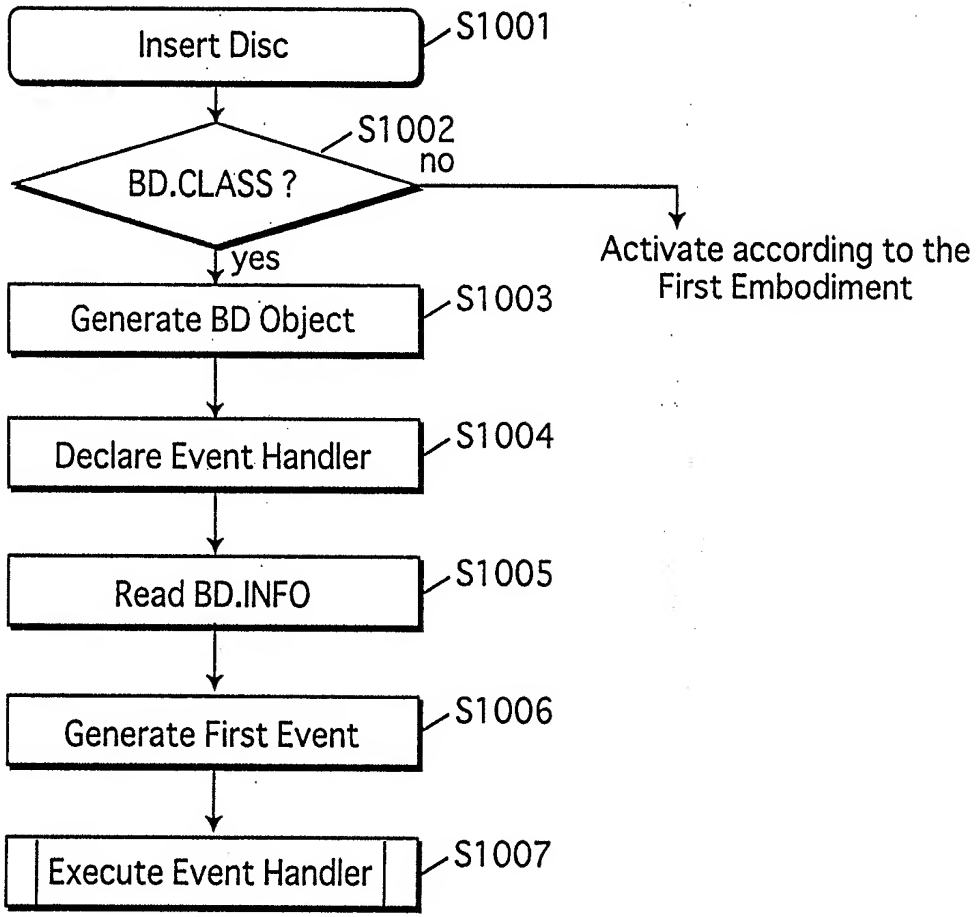

Insert Disc
S1001
S1002
BD.CLASS ?
no
yes
Activate according to the First Embodiment
Generate BD Object
S1003
Declare Event Handler
S1004
Read BD.INFO
S1005
Generate First Event
S1006
Execute Event Handler
S1007

FIG.39

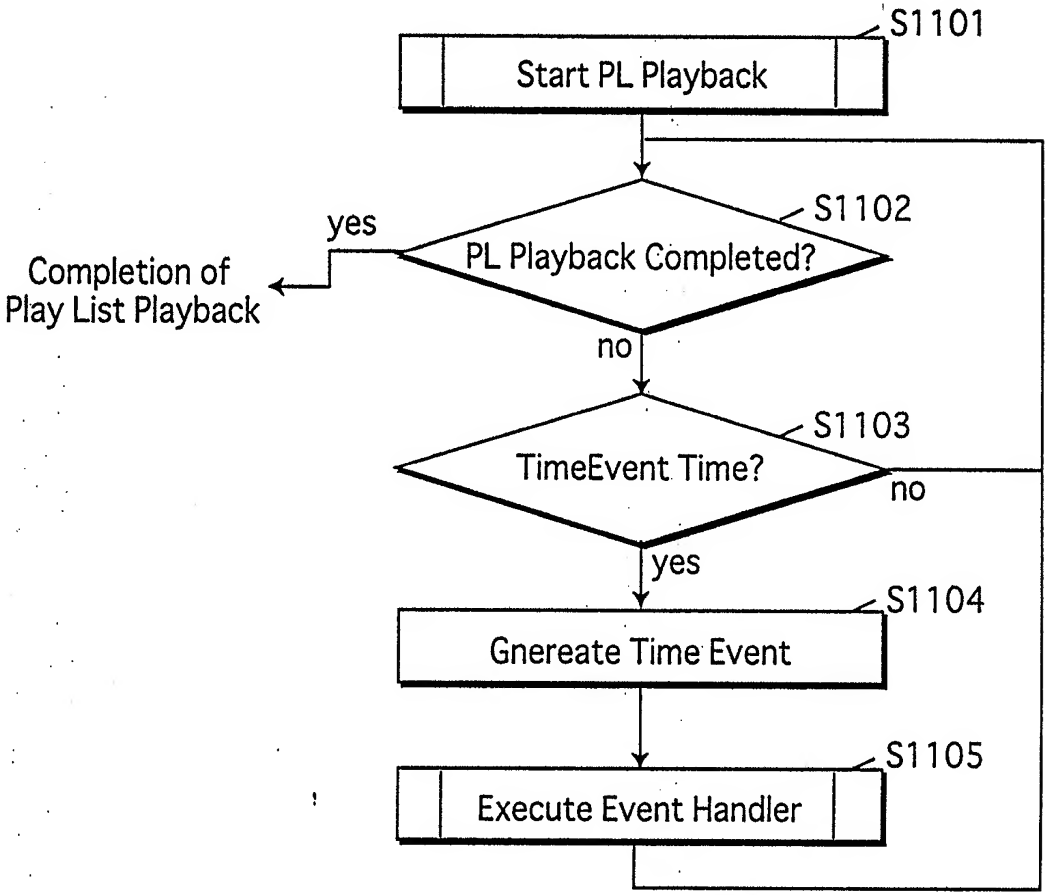

S1101
Start PL Playback
S1102
PL Playback Completed?
yes
Completion of
Play List Playback
no
S1103
TimeEvent Time?
no
yes
S1104
Gnereate Time Event
S1105
Execute Event Handler

FIG.40

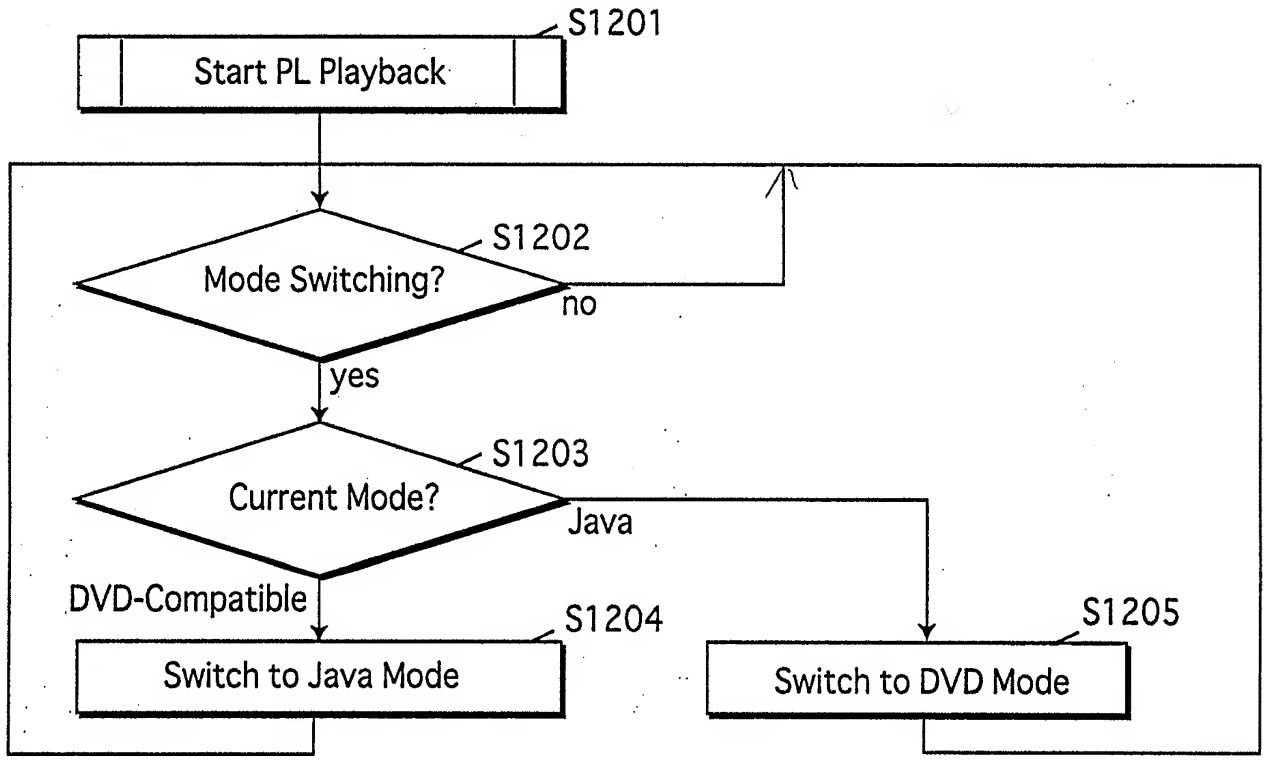

S1201
Start PL Playback
S1202
Mode Switching?
no
yes
S1203
Current Mode?
Java
DVD-Compatible
S1204
Switch to Java Mode
S1205
Switch to DVD Mode

FIG.41

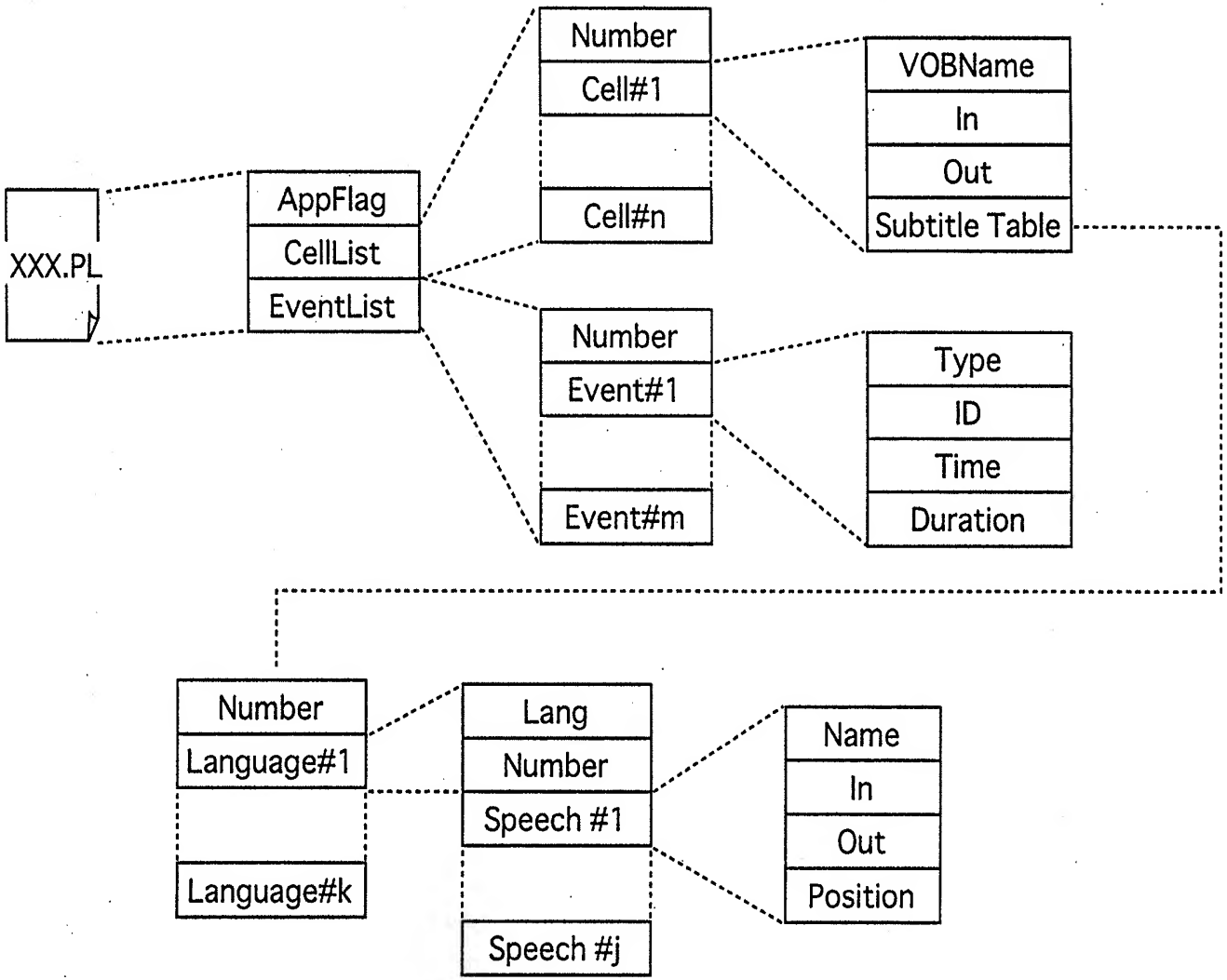

XXX.PL
AppFlag
CellList
EventList
Number
Cell#1
Cell#n
VOBName
In
Out
Subtitle Table
Number
Event#1
Event#m
Type
ID
Time
Duration
Number
Language#1
Language#k
Lang
Number
Speech #1
Speech #j
Name
In
Out
Position

FIG.42

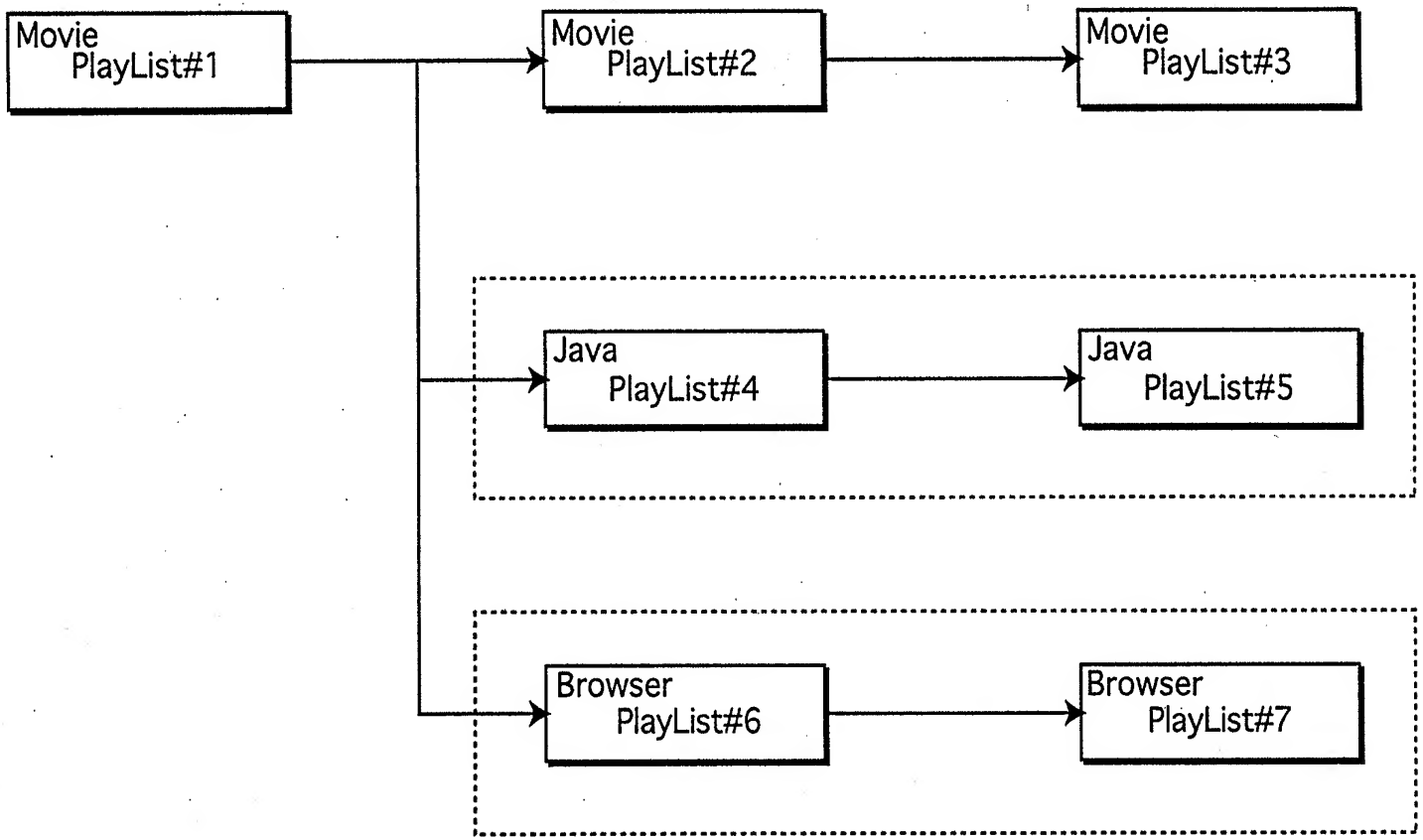

Movie
PlayList#1
Movie
PlayList#2
Movie
PlayList#3
Java
PlayList#4
Java
PlayList#5
Browser
PlayList#6
Browser
PlayList#7

FIG.43

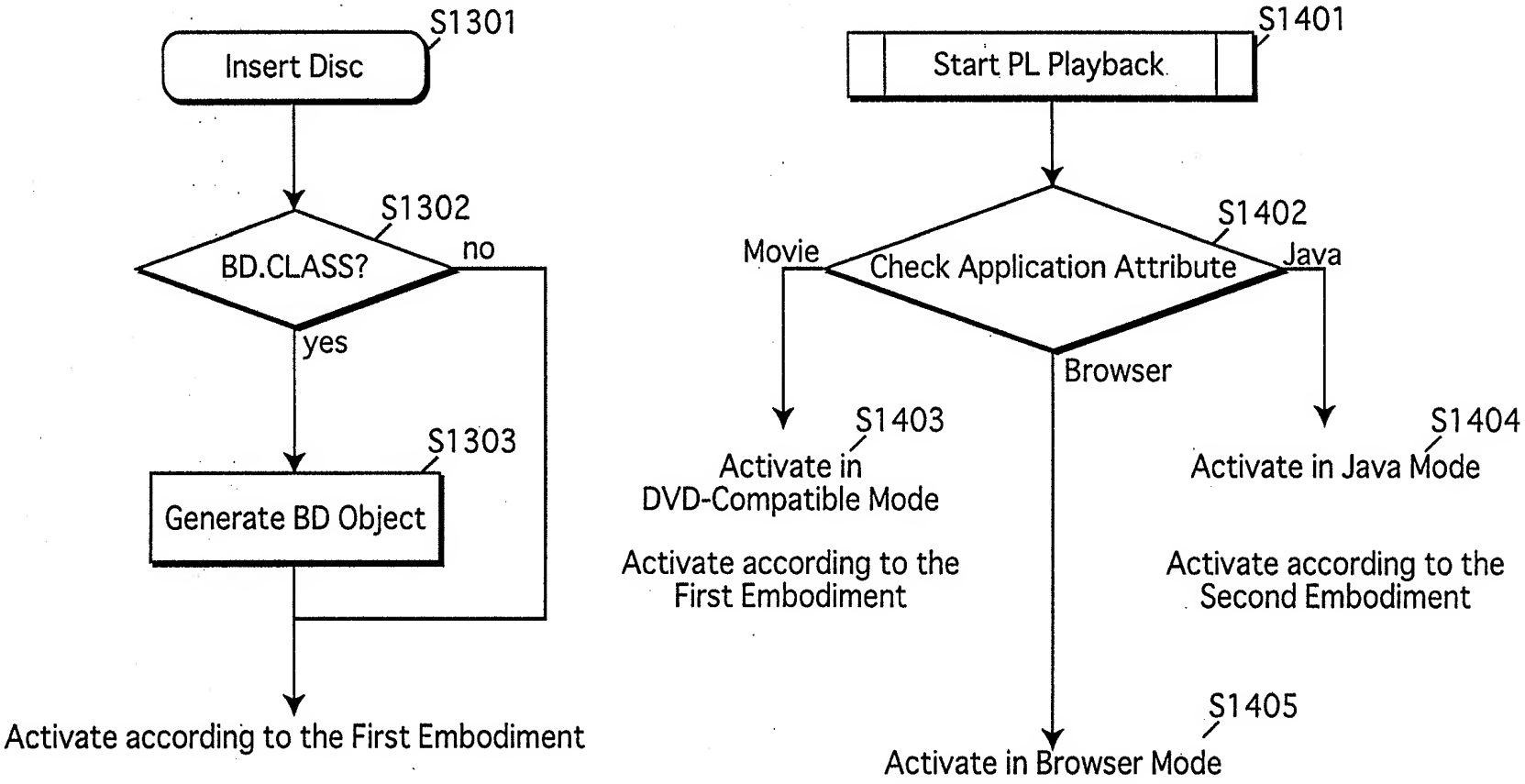

S1301
Insert Disc
S1302
BD.CLASS?
no
yes
S1303
Generate BD Object
Activate according to the First Embodiment
S1401
Start PL Playback
S1402
Movie
Check Application Attribute
Java
Browser
S1403
Activate in DVD-Compatible Mode
Activate according to the First Embodiment
S1404
Activate in Java Mode
Activate according to the Second Embodiment
S1405
Activate in Browser Mode

FIG.44

PLStart	PlayList Starting Event
PLEnd	PlayList Ending Event
Time	Time Event
PauseOn	Pause Starting Event
PauseOff	Pause Ending Event
TPStart	Special Playback Starting Event
TPEnd	Special Playback Ending Event

FIG.45

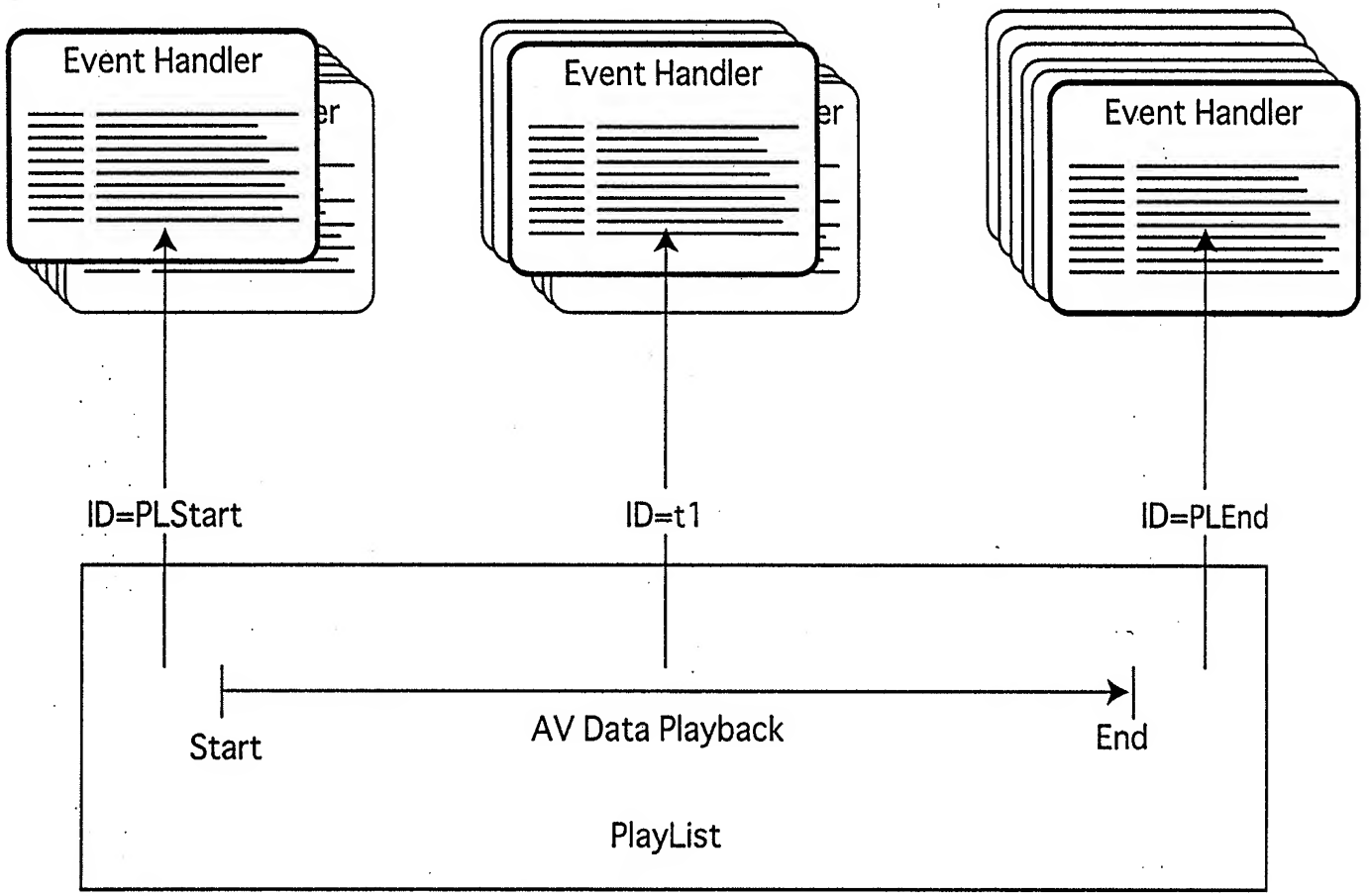

Event Handler
Event Handler
Event Handler
ID=PLStart
ID=t1
ID=PLEnd
Start
AV Data Playback
End
PlayList

FIG.46

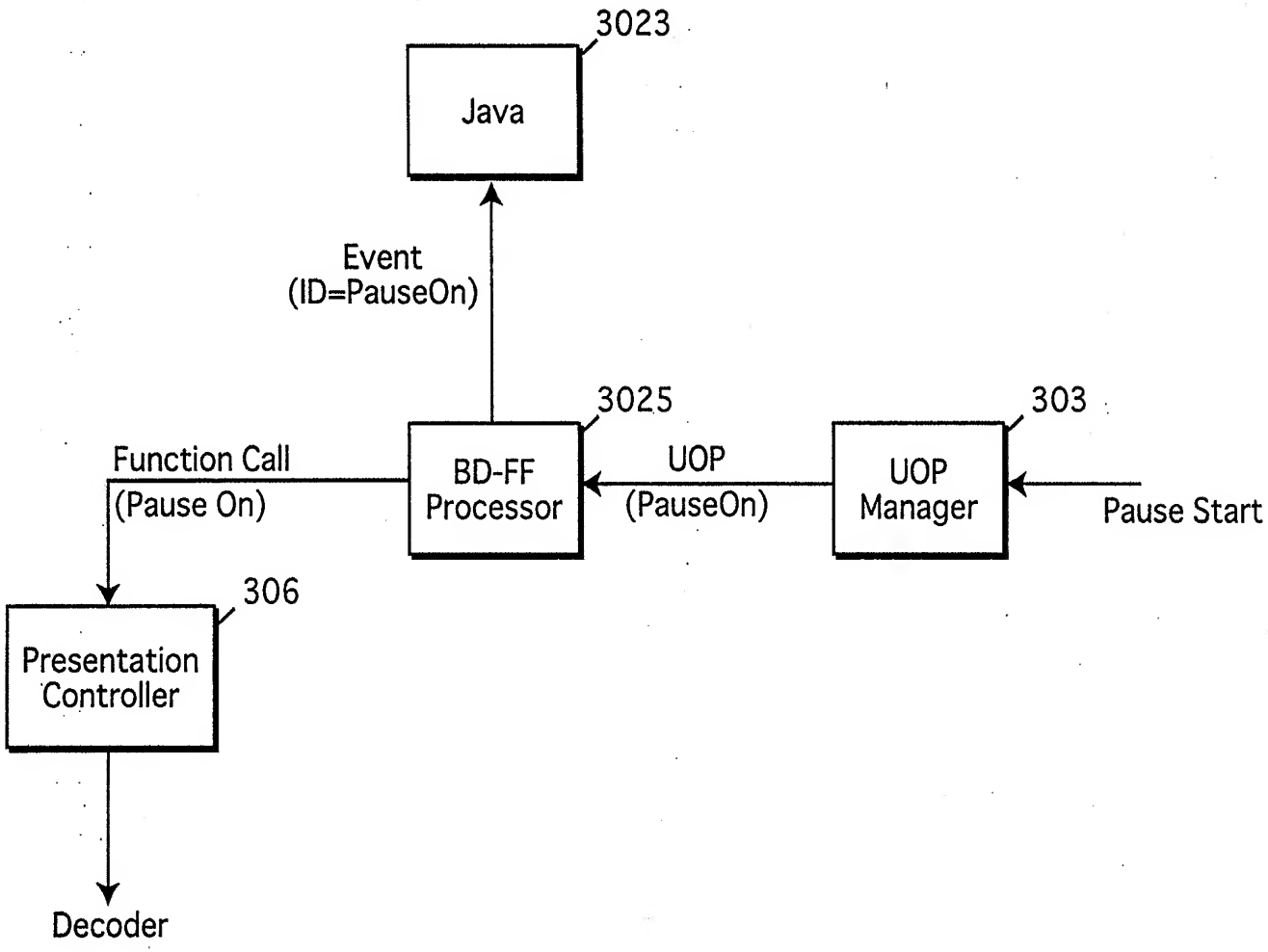
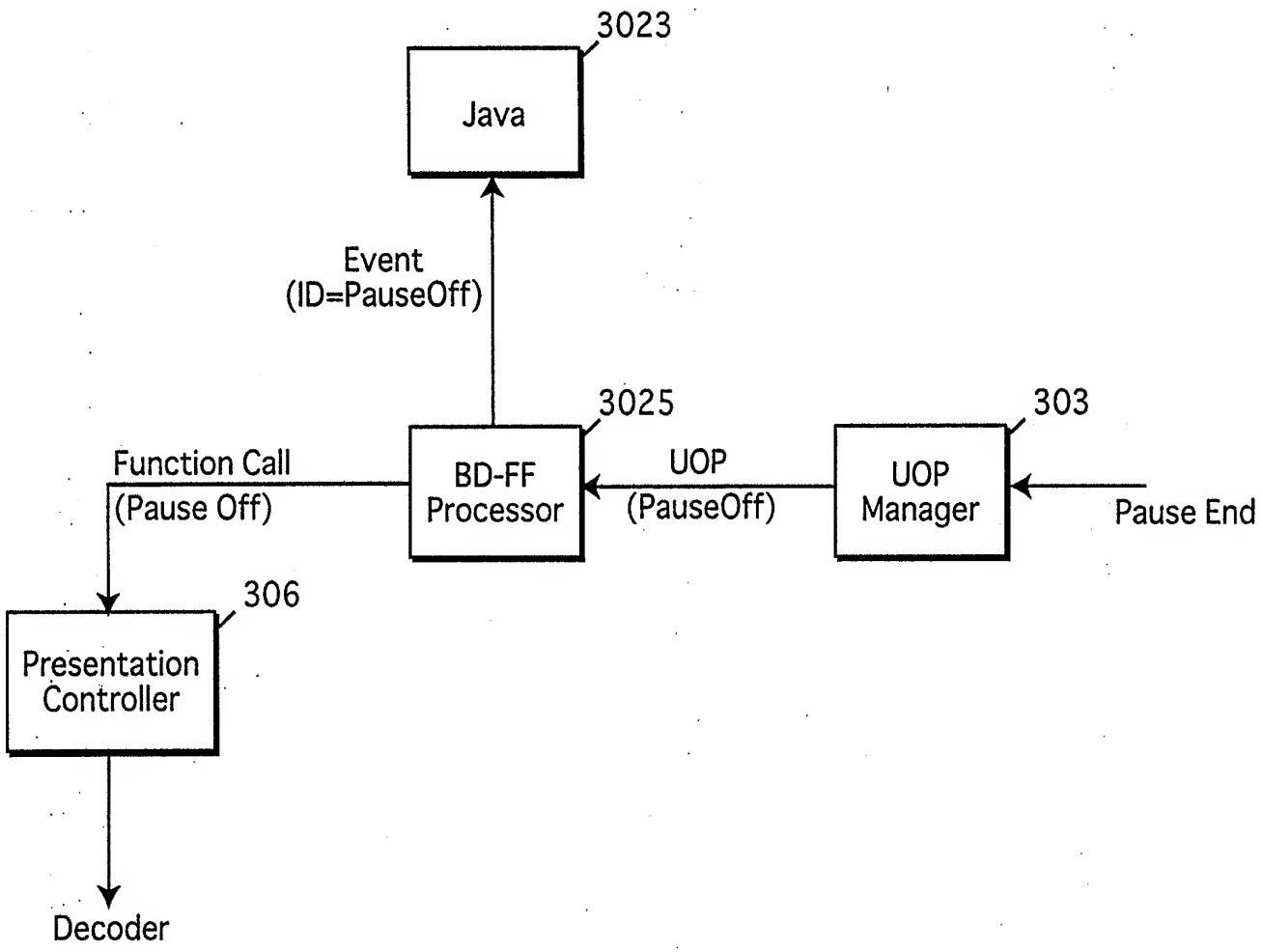

3023
Java
Event
(ID=PauseOn)
3025
BD-FF
Processor
303
UOP
Manager
Pause Start
UOP
(PauseOn)
Function Call
(Pause On)
306
Presentation
Controller
Decoder

FIG.47


3023
Java
Event
(ID=PauseOff)
3025
303
Function Call
(Pause Off)
BD-FF
Processor
UOP
(PauseOff)
UOP
Manager
Pause End
306
Presentation
Controller
Decoder